# A NOVEL VIRTUAL REALITY-BASED SYSTEM FOR REMOTE ENVIRONMENTAL MONITORING AND CONTROL USING AN ACTIVITY MODULATED WIRELESS SENSOR NETWORK

By

Benjamin Montz

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

in

Electrical Engineering

University of Alaska Fairbanks

August 2019

APPROVED:

Dr. Dejan Raskovic, Committee Chair

Dr. Charles E. Mayer, Committee Member

Dr. Denise Thorsen, Committee Member

Dr. Charles E. Mayer, Chair

*Department of Electrical and Computer Engineering*

Dr. Bill Schnabel, Dean

*College of Engineering and Mines*

Dr. Michael Castellini, Dean of the Graduate School

**Abstract**

The ability to monitor and control a home environment remotely has improved considerably in recent years due to improvements in the computational power, reduction in physical size, reduced implementation cost, and widespread use of both wireless sensor networks and smart home systems. This thesis presents a remote environment management system that integrated a custom wireless sensor network that monitored environmental factors in multiple locations, a smart system that controlled those factors, and a virtual reality system that functioned as a remote interface with the environment. The resulting system enabled a user to efficiently interact with a distant environment using an immersive virtual reality experience. The user was able to interact with the remote environments by issuing voice commands, performing hand gestures, and interacting with virtual objects. This type of system has applications in many fields ranging from healthcare to the industrial sector. The case study system that was designed in this thesis monitored and controlled the environments of several rooms in a home.

A novel approach to modulating the activity of the wireless sensor network was implemented in this system. The rate at which the sensor nodes collected and transmitted data was modulated based on the visibility of the virtual objects called VSNs. These virtual sensor nodes displayed the sensor node measurements in virtual reality. This method was expanded upon using a motion prediction algorithm that was used to predict if the virtual sensor nodes were going to be visible to the user. This prediction was then used to modulate the activity of the wireless sensor network. These activity modulation algorithms were used to reduce the power consumption of the wireless sensor network and thus increasing its operational lifespan, while simultaneously reducing unnecessary RF signals in the environment that can interfere with the operation of other wireless systems. These algorithms would be crucial for systems monitoring

complex sensor-rich environments where reducing the data transmitted and extending the system's lifespan was paramount, such as managing the environments of many rooms in a large industrial park or controlling the environments of spacecraft from Mission Control on Earth.

## Table of Contents

## List of Figures

page

## List of Tables

page

## List of Appendices

page

# Chapter 1 Introduction

Virtual reality (VR) is a powerful medium that allows people to interact with environments in an immersive experience. The accessibility and level of immersion of VR systems are mostly constrained by the performance and cost of the necessary computer and VR hardware. In recent years the price and performance of both personal computers and VR systems have improved significantly, bringing VR to a broader market that is forecast to be worth $40 billion by 2020 [1] [2]. While a significant portion of the VR market is entertainment related, there are many other markets being developed. These markets include but are not limited to healthcare, engineering, and education. The markets that take advantage of VR technology are expected to expand as personal computer and VR hardware becomes more powerful, affordable and accessible [3].

The widespread use of wireless sensor networks (WSN) consisting of low-power inexpensive sensor nodes is expected to rival other large milestones in the information revolution. “WSNs provide bridges between the virtual world of information technology and the real physical world.” These networks allow for the real time observation of large-scale real-world environments and events using autonomous inter-device communications [4].

As with VR, smart home systems are becoming more powerful and accessible. Digital assistants that are often included in prevalent smart devices such as smart phones and smart speakers allow for the control of smart home devices using voice commands. These digital assistants can control various smart devices that manipulate a home’s environment. Smart devices can adjust a home’s lighting, temperature, security systems, energy management systems, and electrical appliances [5]. A fully integrated smart home allows for the total control of a home environment from remote locations.

A powerful and synergistic system can be created by combining VR, smart home, and custom WSN technology. These systems introduce the ability of monitoring, controlling, and interacting with remote real-world environments from within a VR environment using VR inputs, hand gestures, and voice commands. Systems with these capabilities have vast potential for use in many applications across several industries. This thesis presents one such system.

The system developed in this thesis makes use of the previously mentioned technology and produced a platform for further research into VR centered WSNs. This system also introduced a novel method of modulating the WSN's activity by adjusting the sensor sampling and RF transmission rates based on the visibility of the virtual visualization components that displayed the sensor data in the VR environment. This method allowed the system to continuously monitor the data for significant events. More importantly, this method allowed for the system to transmit data at a rapid rate that was required for high quality virtual visualization when the visualization components were visible to the user, while also significantly reducing the data sampling and transmission rates when those components were not visible. This method allowed the system to both have a quality VR visualization and be power efficient. This platform allowed for efficient high-fidelity remote monitoring and manipulation of several room environments in a house from within a VR application.

## 1.1 VR Systems

VR allows a user to experience complete immersion in a virtual environment. These environments can be completely fabricated to represent fictional locations. They can also display real locations and events in real time such as live concerts or conferences [6] [7]. In addition to the immersive audio-visual experience, VR often allows for movement around, and the ability to physically interact with, a virtual environment. Motion controllers are used to track hand location

and orientation, and to provide user inputs such as buttons, joysticks and analog triggers. These controllers often include haptic feedback mechanisms that allow for physical feedback interactions in the virtual environment. Motion controllers enable many methods of interacting with virtual objects in the virtual environment ranging from pressing virtual buttons, or pulling virtual levers, to juggling virtual tennis balls, or disarming simulated virtual bombs [8] [9].

The VR experience is not limited to a single user in a virtual environment. Multiple users can share a virtual environment and interact with each other. These multi-user VR experiences have applications in many fields. In the entertainment industry such experiences allow users to work together to fly a simulated virtual spaceship [10]. Corporations have used this functionality to hold meetings in virtual conference rooms that enable communication and collaboration from remote locations across the planet resulting in an experience that felt as if the participants were in the same room [11].

VR systems were introduced into the consumer market as early as 1995. Due to the limitations of the technology available at the time, the experience was not impressive by today's standards and required a significant financial investment [12]. These systems lacked the ability to provide a meaningfully immersive experience. Over the next 23 years, the performance of computer hardware and optics improved significantly while the prices dropped. In the United States, the percent of households that owned a personal computer grew from 22.8% in 1993 to 80% in 2015 [13] [14]. These changes made it possible for the current generation of VR systems to produce incredibly immersive experiences that are accessible to a much larger market. This new level of accessibility has led to a significant increase in investments into new VR technology and research by large companies such as Facebook [15]. The VR Market is projected to go from a compound annual growth rate of $5.4 billion in 2017 to $60.9 billion by 2023 [16].

A large portion of the VR Market is entertainment related, such as video games, live sports, concerts, etc. However, many other industries have started developing VR applications. The VR healthcare market has grown to $976 million by 2017 and is expected to grow to $5.1 billion by 2025 [17]. Similar markets exist in other fields such as VR related engineering applications, and VR based educational tools [18]. With the projected growth and huge potential of VR, it is an important time to conduct research that can expand how VR is used.

## 1.2 WSNs

WSNs allow for the monitoring of environmental data from remote locations. They use embedded systems consisting of very low power sensors and microcontrollers that gather environmental data from remote environments and transmit that data wirelessly across a network to a designated destination for processing. A WSN can consist of many types of nodes, but most consist of sensor nodes that gather and transmit data towards a sink node that aggregates data from the network. Nodes in a WSN tend to consist of a power source, a processor, sensors, and a radio transceiver. The network topologies and communication protocols can vary greatly between WSNs and are often specifically chosen to meet the demands of the specific use case. There are many forms of WSNs that are designed for specific tasks. WSN variants range from personal WSNs that monitor data from sensors worn by an individual to multimedia WSNs that handle images, audio, and video media collected from sensor nodes [19]. The applications for WSNs vary greatly from personal health care monitoring and diagnostics to transportation and logistics [20]. With such a wide range of configurations and applications, it is not surprising that the research field focused on WSN design and implementation is vast. Over 20,000 papers have been written on topics related to WSNs in 2018 [21].

## 1.3 Smart Home Systems

Smart homes are Internet of Things (IoT) systems that give homeowners the ability to monitor and control household devices and home appliances [22]. IoT systems are similar to WSNs, except instead of using a self-contained wireless network, the devices are connected over the Internet [23]. These systems consist of smart devices such as speakers, lights, thermostats, security systems and appliances. Smart homes are usually controlled through dedicated applications or by using voice commands with digital assistants such as Amazon Alexa, Google Assistant, or Apple's Siri [24]. Smart home systems use powerful technology to improve the quality of life of the users.

## 1.4 System Applications

When the features of VR, WSNs, and smart homes are combined, they form a very synergistic system. Systems incorporating these technologies open the doors to many new applications that can vary significantly in scope. A personal WSN that monitors the vital signs of a user, combined with a custom-built VR workout program and a smart home system with temperature and fan control can create an adaptive workout environment. Such a system could keep a user in a physical state that produces ideal workout results without the risk of over-heating or over-exertion. A system that used a medium sized WSN would be capable of monitoring the equipment in a factory using VR as an intuitive and efficient interface. A system that was designed to use a large-scale WSN would have the ability to carry out security monitoring operations covering large complexes of buildings across the country. The security of complexes could be managed with the use of security-focused sensor nodes and smart devices distributed throughout the various rooms of the buildings and a custom-built VR security program. These types of systems have the potential for use in large scale projects involving the power grid, mass transit, transportation of goods, and various industrial applications [25]. The

principles developed in this thesis can be applied directly to augmented reality (AR) systems when they become more capable, accessible, and prevalent in society. The future of these types of systems is promising and just waiting to be developed.

## 1.5 Contributions of Thesis

Some of the main contributions of this thesis are:

1. A platform was developed consisting of a VR program, a WSN, and smart home technology, that enabled further research into the possibilities of such a system.
2. Dynamic sensor data visualizations were developed for VR that presented data coming from a WSN.
3. Virtual components were designed that changed behavior and reacted to variances in sensor node data.
4. An interface was developed that enabled communication between a WSN and a VR program.
5. A contention-free medium access control protocol was designed for the WSN that enabled sensor node activity modulation.
6. An algorithm was developed that predicted if a virtual object in a VR program was going to be visible from a predicted viewpoint determined by the current viewpoint and an average of the user's recent rotational and translational velocities.
7. The main system in this thesis was developed to be capable of adjusting the activity of the WSN based on the visibility of the virtual representation of the sensor node data.

## 1.6 System Overview

The purpose of this thesis was to develop a novel system that combined VR, WSN, and smart home technology so that users could efficiently monitor and manipulate remote environments from inside an immersive VR program. This system was intended to be a proof of concept that could be used as a platform for further research and development. A VR user interface was designed that provided an immersive experience which enabled the user to move around and interact with a virtual environment, while minimizing the factors that can lead to dizziness and nausea, often called VR sickness. Virtual components called virtual sensor nodes (VSN) were developed that displayed data transmitted from, and allow interaction with, the systems sensor nodes. VSNs were designed to provide a constant stream of the most recent sensor data from the sensor nodes in both text and graphical representations. The VSNs also provided the interface that allowed the user to interact with the sensor nodes. These interactions consisted of pressing virtual buttons or performing specific hand gestures in VR using motion controllers. While these events were limited in scope, they are just a small example of what is possible. The system made use of an algorithm that detected if the VSNs were visible or if they were expected to be visible based on the current VR motion of the user. This algorithm was used to adjust the sensor node's data collection and transmission rates in order to extend the lifespan of the WSN while reducing the data throughput of the system. The communication protocols used in this system were designed specifically to meet the requirements of this project. The sensor nodes were designed to collect environmental data including ambient temperature and light intensities, the physical orientation of the sensor node and the state of push buttons. These buttons were used as a stand in for reed switches that could detect if a window or a door were open or closed. The sensor nodes also included an RGB LED and a buzzer that were used to react to signals coming from the VSN. A user interface was also designed that allowed for the

setup and monitoring of sensor nodes. This user interface displayed the most recent data from both the sensor node and the corresponding VSN. Smart Home technology was also incorporated into the system so that the user could give voice commands while using the VR program that would adjust the brightness of the smart lights located in the same rooms as the sensor nodes. While WSNs and smart home technologies provide a window between the world of virtual information technology and reality, the system designed in this thesis opens that window and allows the user to reach through it.

## 1.7 Thesis Organization

Chapter 2 presents an overview of VR, WSNs, and smart home research and technology.

Chapter 3 describes the architectural layout of the system designed in this thesis. This chapter provides an overview for how the components of the system work and function.

Chapter 4 describes the hardware implementation of the system. The hardware setup and capabilities of all the systems are explored here.

Chapter 5 gives an account of the software implementation of the system from the configuration sensor and sink node to the implementation of the VR program.

Chapter 6 describes the algorithm that was used to modulate the activity of the WSN.

Chapter 7 presents the results of tests that were performed to demonstrate the functionality of system.

Chapter 8 provides a conclusion and direction for future research that can improve similar systems in the future.

# Chapter 2 Literature Review

The current growth in the adoption of VR technologies has led to significant capability and performance improvements in both VR related hardware and software. New VR systems have been brought to the market in recent years with new and improved models being released regularly. Significant research into VR hardware and software has been conducted to improve the VR experience. The wide focus of the research ranges from VR gloves that track individual finger movement, to VR programs that monitor sensor networks [26]. Research is also being conducted across the wide field of WSN design. These areas include but are not limited to network topologies, medium access control (MAC) protocols, energy-efficiency, and applications [4]. The smart home technology sector has also had rapid growth in recent years, from an influx of new smart devices to advancements in digital assistant capabilities. This chapter covers the current research in these areas as they relate to this thesis.

## 2.1 WSNs


*Figure 1: WSN Applications*

With the reductions in manufacturing costs and advancements in various fields of embedded systems, networked embedded systems have become ubiquitous throughout society. Many forms of networked embedded systems exist, but a powerful subset of these systems called WSNs has been growing in capability and popularity. They appear in domains ranging from office buildings to industrial environments as Figure 1 shows. The classic roll of these systems has often been focused on environmental monitoring, but recently there has been a focus on environmental control [27]. WSNs consist of several embedded system-based sensor devices that are spread across a geographical area. These devices often have a set of tasks that are required of them that include sensing data, processing that data, and communicating with other nodes in the network. The intention of these systems is usually to gather data from the sensor nodes and transmit that data to centralized devices with higher processing power for further data processing where the recorded values can be displayed and acted upon [28]. While every system application has its unique challenges, almost all WSNs share a set of common challenges. The most common of these challenges include energy efficiency, wireless signal interference, and data management. Energy efficiency can be managed using low power hardware as well as clever software design that reduces the active time of the embedded systems in the network. Data and wireless signal interference management can be accomplished in many ways depending on the complexity of the network [29]. Selection of the best hardware, network topology, and communication protocols for a given application play a large part in managing these challenges.

### 2.1.1 WSN Nodes

A WSN is a system that consists of many embedded systems called nodes. Nodes can be designed with various capabilities and can be placed in several categories but are often divided into two main types; sensor nodes that collect and sometimes process environmental data, and sink nodes that gather, further process, and pass on sensor node data to end users. The sensor nodes can be designed to incorporate a vast array of sensors ranging from ambient temperature sensors to high definition cameras [4]. It is often required that sink nodes manage the WSN's communication network.

### 2.1.2 Power Efficiency Techniques

Significant efforts have been made to develop techniques that improve the power efficiency of WSNs. Most WSN nodes are powered by batteries, so an improvement in power efficiency leads to the reduced cost and lifespan of the entire network. Many methods can be used to increase the power efficiency of a WSN. The most common and straight forward method is to use low power hardware and reduce the activity levels of nodes in the WSN. Reducing the activity level of a node is accomplished by setting the node's hardware components into a low-power mode until they are needed, and minimizing the time spent in an active mode. In addition to this, selecting the proper RF settings in the transceiver can have a dramatic impact on the power efficiency of a WSN. RF data rates, carrier frequency, and RF output power all have significant effects on the power consumption of a WSN node [30]. More complicated ways to increase the power efficiency of a WSN include selecting proper topology and communication protocols for the network. This reduces the number of sensor nodes that need to be active, as well as the time that the sensor nodes need to be in active mode [31]. There are numerous network topologies and communication protocols to choose from, but it turns out that in a normal

RF environment, using a single-hop routing topology is almost always more power efficient than multi-hop routing topologies [32].

### 2.1.3 Network Topologies

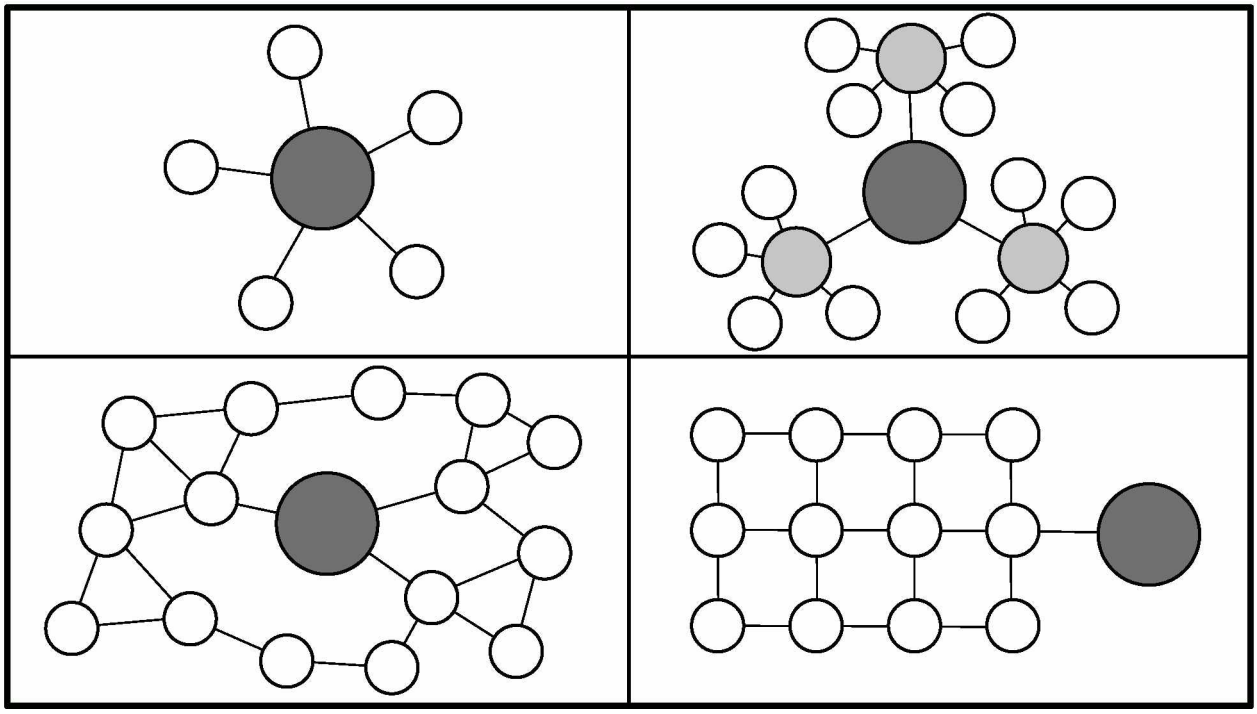

*Figure 2: Network Topologies. (Top Left) Single-Hop Star, (Top Right) Two-Tier Hierarchical Cluster, (Bottom Left) Multi-Hop Mesh, (Bottom Right) Multi-Hop Grid*

There are many ways to configure the network topology in a WSN. Each topology has benefits and drawbacks. Some of the most common network topologies include single-hop star, multi-hop mesh, multi-hop grid, and multi-tier hierarchical cluster. Graphical representations of these topologies can be seen in Figure 2. Single-hop star topology is one of the simplest topologies. In such a network, the sensor nodes send their collected data directly to a centralized sink node for further processing. While this is a simple topology with many benefits, it has poor scalability and robustness. For larger and more complex networks, multi-hop mesh and grid topologies are useful as they allow for the sensor data to be transmitted to the centralized sink node indirectly through other sensor nodes. They typically provide multiple paths for data, which

increases reliability. These networks are very useful but make the system more complex to design and are often less power efficient. Two-tier hierarchical cluster topology is a powerful network configuration where the network is separated into several clusters made up of several sensor nodes and a centralized node called the cluster-head. The sensor nodes transmit their data to their cluster head for initial processing. The cluster head then uses a secondary network to transmit the clusters data to a secondary centralized node for further data processing, often using a separate transceiver and different RF bands. This topology can be useful for splitting a large network into distinct zones where data aggregation and processing can be handled locally. As with the multi-hop mesh and grid topologies, the multi-tier cluster topology design increases the complexity while reducing the power efficiency of the system. Careful consideration must be taken when deciding on a systems network topology [4].

### 2.1.4 Media Access Control Protocols

In order to maintain reliable communication channels and an energy efficient design, it is necessary for a WSN to maintain control of the access to RF channels. With poor RF channel control, a system encounters a host of issues including signal interference and lost data packets. These issues lead to unnecessary activity in the network, reducing the power efficiency of the system. MAC protocol design is a large field of research with many potential solutions that may work for an assortment of applications. MAC protocols are often designed for a range of WSN types, but not for any specific application. In general, WSN-focused MAC protocols can be divided into two categories, contention-based, and contention-free. Contention-based protocols require that a node listens to the RF channel before transmitting data to ensure that it is not being used. If the channel is busy, the system must wait before trying again. While these protocols offer robustness and scalability, they can cause nodes to be active for an unacceptable duration in

systems with heavy traffic. Contention-free MAC protocols use various methods to manage access to the systems RF channel. These methods do not require the nodes to check for busy channels before transmitting data. One of the most commonly used methods for controlling RF channel access in contention-free MAC protocols is time scheduling. Systems using time scheduling techniques require a synchronization event to ensure all the nodes are synchronized. Synchronization is often performed by the transmission of a synchronization packet by a node that is within range of the other nodes in the system. This synchronization packet is transmitted at a regular interval in order to keep the network continuously synchronized. Once the network is in sync, the RF channel is divided into periodic frames of fixed duration. These frames are then further divided into time-slots that are assigned to specific nodes, so that nodes can only transmit during their designated time-slots. These methods help the system maintain a higher power efficiency but run into trouble when it comes to scalability and channel utilization when used with low data rate networks [33]. There are many design choices involved in selecting a MAC protocol. Careful consideration needs to be taken when selecting the protocol used in a system to ensure optimal performance.

## 2.2 Smart Home Technology

A smart home is "a dwelling incorporating a communications network that connects the key electrical appliances and services, and allows them to be remotely controlled, monitored or accessed". These systems need an internal network, some form of intelligent control, and use some form of automation [34]. Smart devices are the main components of smart homes that can be monitored and controlled. Numerous new consumer smart devices have entered the market in recent years. The various smart devices can often be accessed from any location with a cellular or Wi-Fi signal using an application on a smart phone. While the smart home system gives the

homeowner the power to control their home from almost anywhere, it does come with some challenges, such as access control and privacy [35]. The same technology that enables smart homes to be so impactful also allows for more complex systems, such as smart health systems, smart factories, and smart cities [36].

### 2.2.1 Smart Devices

As mentioned earlier, smart devices are components that are controlled through a smart home application in a smart home or through voice commands given to a digital assistant. There are many types of smart devices, and more are being developed all the time. The most common consumer smart devices are: [37]

- Lighting Solutions
- Thermostats
- Air Conditioners
- Security Systems
- Motion Sensors
- Door Locks
- Cameras
- Electrical Plugs
- Speakers
- Televisions
- Vacuums
- Kitchen Appliances
- Lawn Sprinklers

### 2.2.2 Digital Assistants

Digital assistants are artificial intelligence-based voice assistants that recognize and understand verbal commands. These assistants are included in digital assistant enabled devices like smart speakers and smart phones. They use natural spoken language to communicate with users and are capable of perform various tasks ranging from reading the news or weather forecast, sport information, ordering products online, searching the internet, and most importantly interacting with smart devices [38]. An example of an interaction with a digital assistant could go as follows:

User: "Hey Google, turn off the lights."

Google Assistant: "Sure, turning off four lights."

User: "Hey Google, are my lights on?"

Google Assistant: "The Light 1, The Light 2, The Light 3, and The Light 4 are off."

## 2.3 VR

For decades, researchers have recognized the potential of VR as powerful human-computer interface. While there has been some disagreement on what specifically defines VR, the generally accepted definition is as follows; VR is a simulation that uses computer generated graphics to create a dynamic realistic-looking world that responds to the user's input. The three I's of VR need to be considered to produce a high-quality VR experience. These three I's are Interaction, Immersion, and Imagination [39]. Interaction is the ability for the user to interact with the virtual environment. Immersion is when a user begins to feel as if they exist in the virtual environment and can sometimes lead to temporarily forgetting about their physical surroundings. Imagination is required in designing the VR experience to be an easy to use and efficient tool, as everything in the virtual environment should be created with VR immersion in

mind. VR was considered to have arrived in a "barely working" state in 1999. However, over the next two decades incredible advancements in various VR fields have been made, both by the industrial and academic communities. While the popular view of VR is focused on the entertainment sector, it is viewed as a powerful tool with significant potential in various industrial sectors. With the countless improvements made over the last two decades in both hardware and software, many professionals agree that VR now has fully arrived, is stable, mature, and usable. VR is currently being used in various industries to enhance decision making and innovation [40].

### 2.3.1 VR Immersion

Immersion is one of the most important aspects of a VR experience. It can be defined as "a sensation of being in an environment." This can be divided into two main categories, mental immersion and physical immersion. Mental immersion is a state of deep engagement with a suspension of disbelief. This kind of immersion is common in various kinds of media from books to movies. Physical immersion is when the user can physically interact with a medium. VR technology is used to provide a synthetic stimulus to the user's sense of hearing, sight, and touch. With the use of both mental and physical immersion, VR technology allows for a user to explore a simulated reality that approaches physical reality [41]. When a VR program is designed, every choice should consider if it will take users out of the immersed state of mind. While many factors can break immersion in VR, the most common things that break it are reduced frame rates, screen jitter, loss of head or hand tracking, and poor user interface design.

### 2.3.2 Consumer VR Hardware

The consumer market has seen a large number of VR headsets enter the market over the last few years, as can be seen in Table 1 [42] [43] [44] [45]. These headsets fall into three main categories: tethered, standalone, and mobile. Tethered headsets connect to an external computer that handles the processing involved in generating a VR experience. These VR systems produce the highest quality experiences by taking advantage of the available higher processing power and more advanced sensors and controllers. Standalone headsets are all in one devices that do not require any external components to produce VR experiences. They are considered mid-range devices that produce mediocre but consistent experiences. Mobile headsets use smart phones to produce a low-end VR experience that is often used to introduce people to VR.

*Table 1: Current Consumer VR Hardware*

| Company | Product | Headset Type | Resolution (per eye) | Refresh Rate (Hz) | Sensors | Controls | Price (USD) |
|---|---|---|---|---|---|---|---|
| Oculus | Rift | Tethered | 1080x1200 | 90 | Motion, external vision positioning | Oculus Touch, Xbox One Gamepad | 699.99 |
| Oculus | Rift S | Tethered | 1280x1440 | 80 | Motion, Inside-out five camera tracking | Oculus Rift S Controllers | 399.00 |
| HTC | Vive | Tethered | 1080x1200 | 90 | Motion, camera, external motion tracking | HTC Vive motion controllers | 499.00 |
| HTC | Vive Pro | Tethered | 1440x1600 | 90 | Motion, camera, external motion tracking | HTC Vive motion controllers | 798.97 |
| Sony | Playstation VR | Tethered | 960x1090 | 120 | Motion, external vision positioning | DualShock 4, PlayStation Move | 219.95 |
| Oculus | Go | Standalone | 1280x1440 | 60 | Accelerometer, gyroscope | Oculus Go Controller | 199.00 |
| Oculus | Quest | Standalone | 1440x1600 | 72 | Motion, Inside-out four camera tracking | Oculus Quest Controllers | 399.00 |
| Samsung | Gear VR | Mobile | Phone dependent | Phone dependent | Motion | Handheld remote, touchpad on headset | 129.99 |
| Google | Daydream View | Mobile | Phone dependent | Phone dependent | Motion | Daydream Controller | 99.00 |
| Google | Cardboard | Mobile | Phone dependent | Phone dependent | Motion | None | 15.00 |

## 2.4 WSNs with Augmented and VR Visualization

A system that uses a WSN to collect environmental data can be powerful, but the way in which the data is displayed can be just as important to the functionality of the system. Over the years, researchers have developed various VR and AR based systems for interacting with WSNs. In 2008, an AR interface called SensAR was developed that allowed a user to visualize text and 3D graphics representing temperature and sound levels gathered from a WSN. This system used a handheld AR capable device to display the visualizations. While the system made important advancements, the computational power and performance of the hardware at the time was limited and provided a lower sense of immersion than current systems allow for. Figure 3 shows the SensAR interface in action [46].

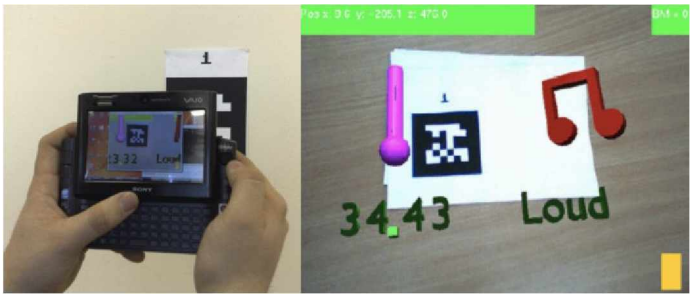


*Figure 3: SensAR Interface, (Left) AR Display Device, (Right) AR Viewpoint [46]*

In 2015, a more advanced AR based WSN visualization system was created. This system used AR technology to visualize and manage a variety of wireless networks. Sensor node data was displayed in AR, as well as various wireless network related data such as connections, data rates, and addresses. As with the SensAR, the data was visualized using text and 3D graphics viewed through an AR interface. This interface can be seen in Figure 4 [47].

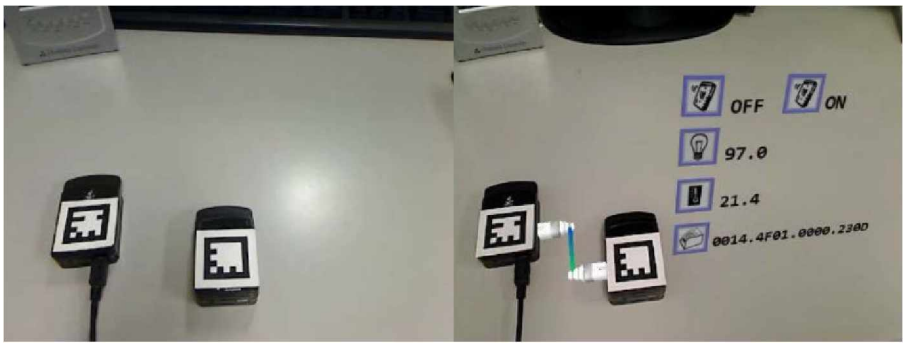


*Figure 4: Advanced AR Visualization Interface, (Left) No AR Overlay, (Right) With AR Overlay [47]*

VR systems have also been designed for visualizing WSN data in recent years. In 2016, a company called Eye Create Worlds, working with leading IoT companies, created an interface for displaying data coming from IoT sensors that were designed for the transportation industry [48]. This interface allowed the user to visualize the sensor data from multiple viewpoints as shown in Figure 5.

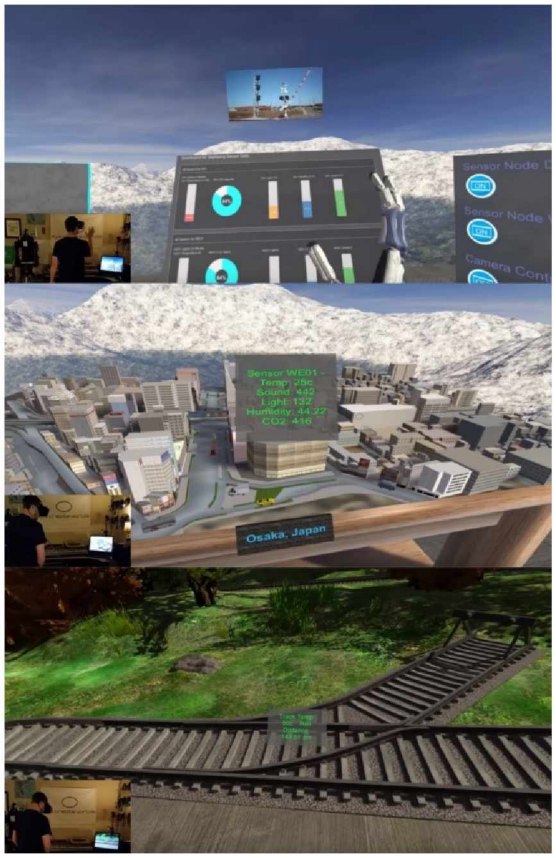


*Figure 5: Eye Create Worlds VR IoT Visualization System,*
*(Top) Sensor Data View, (Middle) City Top Down View, (Bottom) Ground Level View [48]*

## Chapter 3 System Architecture

This chapter discusses the architectural outline of the prototype home management VR system (HMVR) designed in this thesis. This system was designed as a proof of concept. The same design principles could be used in many other applications, such as the medical or industrial sectors. The HMVR system was designed to enable a user to remotely monitor and control the environments throughout their home from inside a VR program using voice commands, virtual gestures, and interaction with virtual objects. This chapter describes how the system was laid out as well as its functionality. Chapters 4, 5, and 6 provide an in-depth investigation into the hardware and software used.

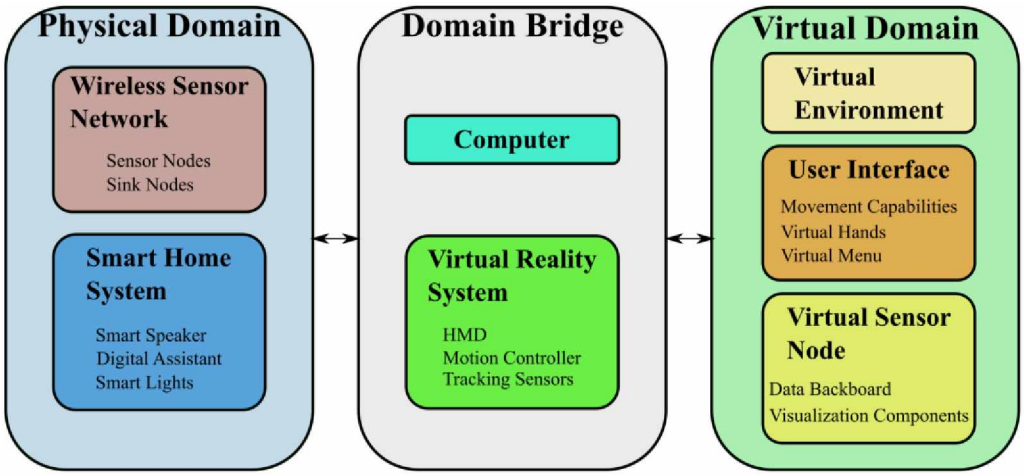


*Figure 6: HMVR System Architecture Outline*

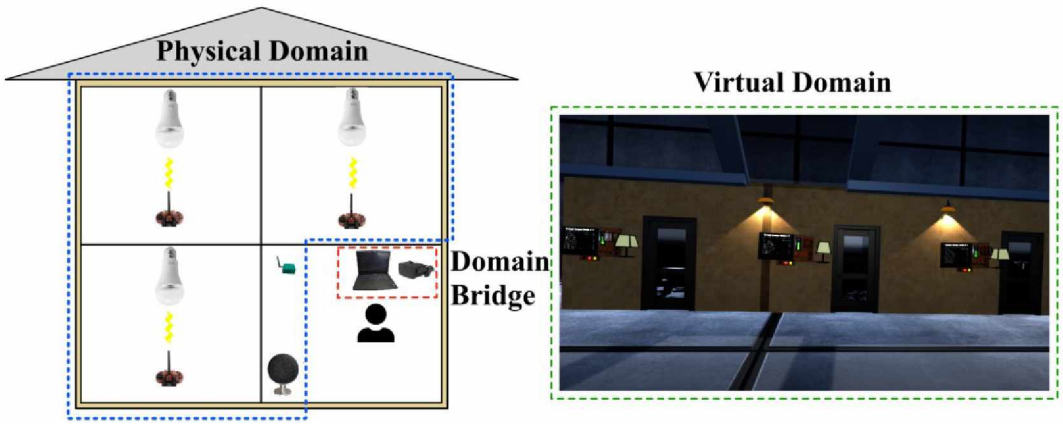


*Figure 7: HMVR System Architecture Diagram*

The HMVR system was divided into three sections, as shown in Figure 6 and Figure 7. The first section consisted of the components in the physical domain. This domain included the WSN and the smart home system. The WSN consisted of three sensor nodes and one sink node, while the smart home system was made up of one digital assistant enabled smart speaker and three smart lights. The second section consisted of the components that bridge the physical domain with the virtual domain: A computer and a VR system. The VR system consisted of a tethered head mounted display (HMD), motion controllers, and positional tracking sensors. The third section included the components in the virtual domain. This section was made up of the virtual environments, the user interface, the VSNs, and the interactable components that reacted to values stored in the VSNs. The VSNs consisted of a data backboard, and several visualization components that presented sensor node data using a 3D graphical representation. The system was designed so that each senor node would be represented by a single VSN. Node IDs were used to indicate which sensor node corresponded with which VSN. For example, senor node #1 would correspond with VSN #1.

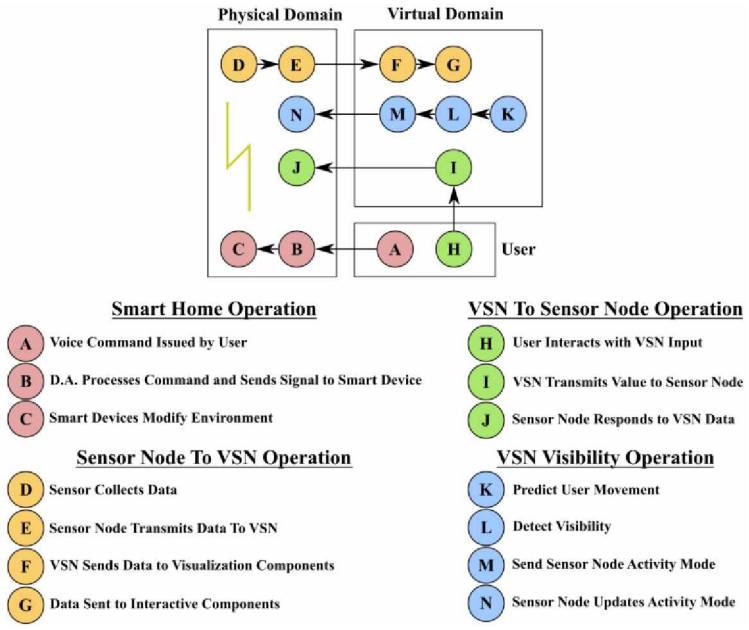


*Figure 8: HMVR Concurrent Operation Diagram*

The functionality of the HMVR system can best be summarized by four concurrent operations as shown in Figure 8. The "Smart Home Operation" took in the user's voice commands and adjusted the brightness of the smart lights. This operation began when the user gave a voice command to the digital assistant in the smart speaker. The digital assistant then processed the command and sent the appropriate signal to the smart lights. Once the smart lights received the signal, they adjusted their brightness accordingly. A single voice command could be used to modify the brightness of multiple smart lights. While this operation did not affect the behavior of the sensor node, it did adjust the measured brightness values. The "Sensor Node to VSN Operation" was used to transfer data from the sensor node to the corresponding VSN. It began with the sensor node receiving new values from the sensors. The new data was then transmitted to the sink node over the wireless network, where it was further transmitted to the VR program running on the computer. The data was then sent to the designated VSN that shared the same ID as the sensor node. Once the VSN had the data, the new values were displayed numerically on the VSN's data backboard. The data was then passed along even further to the virtual visualization components. These components visualized the new data and finally passed the values along to any interactive components that they were attached to. The virtual interactive components were then able to react to the new data. These interactive components allowed for the values coming from the sensor nodes to modify the virtual environment. The "VSN to Sensor Node Operation" was used to transfer signals from the VSN back to its corresponding sensor node. It started with the user, by means of the VR System, interacting with the inputs on the VSN. The VSN then transmitted a signal, corresponding to the VSN input, to the WSN's sink node, which was then transmitted to the corresponding sensor node. Once the sensor node received the signal it either adjusted the color emitted by an RGB LED or produced a series of

beeps from the buzzer based on the received signal. These outputs were used as a stand-in to demonstrate how interactions with the VSN could be used to modify the behavior of the WSN. The "VSN Visibility Operation" was used to detect if VSNs were visible or were expected to be visible to the user in VR, and then to modulate the activity of the corresponding sensor node accordingly. This operation began by making a prediction of the location and orientation of the user based on current velocities. The VSNs were then evaluated to see if they were visible to the user, or if they were expected to be soon. If the visibility status of the VSN changed, a signal was sent to the WSN's sink node to change the activity level of the corresponding sensor node. The sink node then transmitted a signal to the sensor node instructing it to change its activity level. Once the sensor node received this signal, it modified its activity level accordingly. This operation is discussed in detail in Chapter 6.

## 3.1 WSN

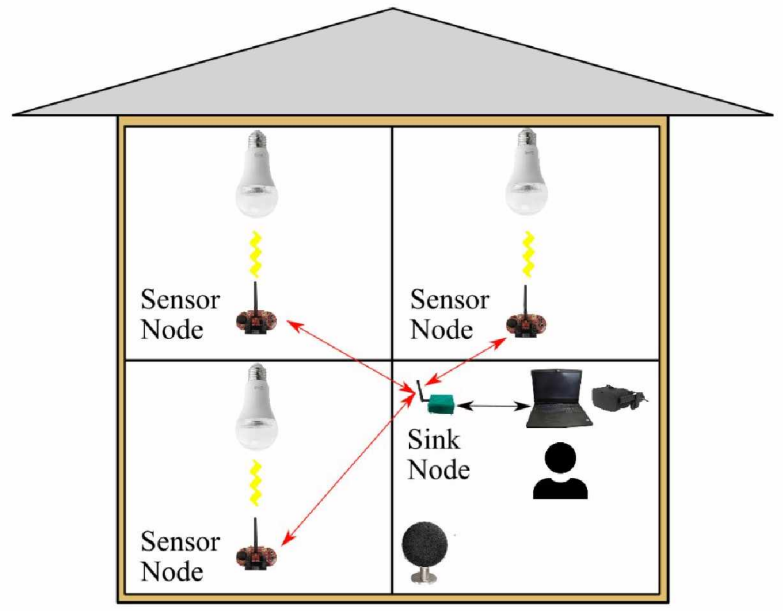


*Figure 9: HMVR WSN Diagram*

The WSN consisted of three sensor nodes and one sink node connected through a 2.4 GHz wireless network using a simple single-hop star topology. The system was designed so that

the sensor nodes could be placed in various rooms of a building while communicating with the sink node in a room designated for VR use, as shown in Figure 9. The sensor nodes monitored environmental values in their respective rooms and transmitted that data to the VSNs inside the VR program. The communication protocols used in the wireless network, and to communicate between the sink node and the VR program were custom designed and are described in detail in Chapter 5.

### 3.1.1 Sensor Nodes

*Figure 10: HMVR Sensor Node*

Figure 10 shows a sensor node that was used in the HMVR system. The sensor nodes collected and processed environmental data from their local environments and transmit that data, by way of the sink node, to the corresponding VSN. Three sensors were used to collect 3-axis acceleration, ambient light, and ambient temperature data. Two buttons were also used as stand-in inputs to represent reed switches connected to a door and a window. An RGB LED and a Buzzer were used to react to signals coming from the sensor nodes corresponding VSN. To setup

a sensor node, a console-based setup interface was used. The sensor nodes used pre-fabricated experimenter boards that contained several components that were not used or were not power efficient. When possible, these components were turned off or set into their lowest power mode to maintain power efficiency. A custom HMVR sensor node board could be produced in the future to greatly reduce the size and power consumption of the device.

### 3.1.2 Sink Node

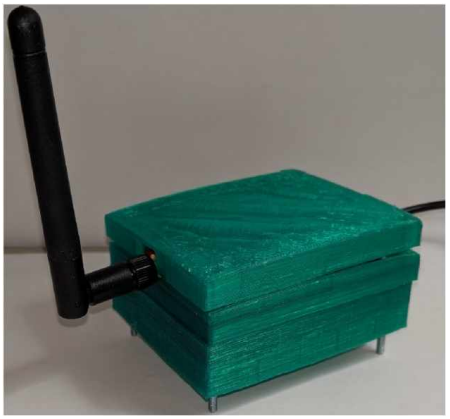

*Figure 11: HMVR Sink Node*

Figure 11 shows the WSNs sink node. The sink node was responsible for handling the communication between all the sensor nodes and the VR program. It was also responsible for maintaining synchronization of the WSN and detecting if a sensor node had lost its connection with the WSN.

### 3.1.3 Wireless Network

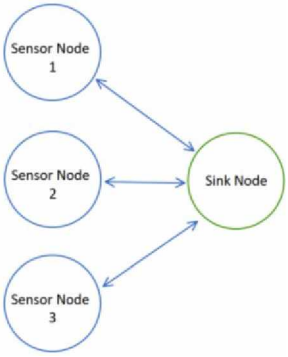


*Figure 12: WSN Topology*

The wireless network consisted of 3 sensor nodes and a sink node in a single-hop star topology which can be seen in Figure 12. The network used 2.4 GHz transceivers communicating with signal strengths of 0 dBm and over the air data rates of 2 Mbps. The network used a custom contention-free MAC protocol that is described in Chapter 5.

## 3.2 Smart Home System

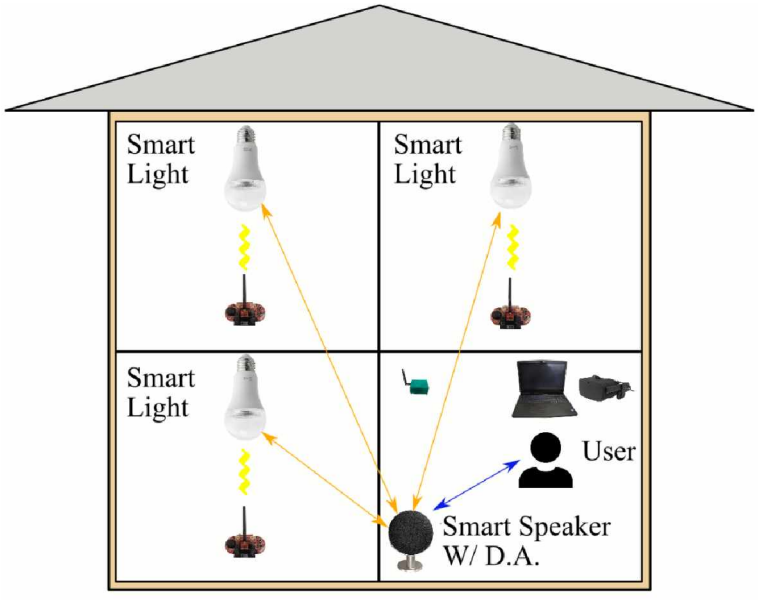


*Figure 13: HMVR Smart Home System Diagram*

The smart home system consisted of a digital assistant enabled smart speaker and three smart lights with adjustable brightness levels. This setup is diagramed in Figure 13. Voice

commands given to the digital assistant were used to adjust the brightness of any of the smart lights. A single voice command could be worded to control the brightness of one, two, or all three of the smart lights at once. Once a voice command was given, the digital assistant processed it into data that was sent to a cloud-based server for processing the intent of the command. Once the command was understood, a process that was specific to the brand and type of smart light was carried out to adjust the brightness. While the smart home system does not adjust how the WSN behaved, it did adjust the values measured by it. The smart home system working in tandem with the custom WSN allowed the user to both monitor and manipulate the environments from a remote location using the VR interface.

## 3.3 Computer

The computer that was used to run the VR system was an off the shelf laptop running the Windows 10 operating system. It was used because it was recommended by the VR hardware's manufacturer indicating it met the requirements to enable the VR hardware to produce an excellent VR experience. The computer was responsible for running the VR program and needed a powerful enough processor and graphics card to provide a high and stable frame rate to maintain a strong sense of immersion. The hardware specifications of the laptop used are discussed in Chapter 4.

## 3.4 VR System

The VR system consisted of the VR hardware and the VR program running on the computer. The VR system is responsible for producing the VR experience for the user.

### 3.4.1 VR Hardware

This system used an off the shelf VR hardware set that consisted of one HMD that enabled the user to see and hear the virtual environment, two motion controllers that allowed the

user to move their hands around in order to interact with the virtual environment, and two motion tracking sensors that tracked the users head and hand positions and orientations. The hardware is described in greater detail in Chapter 4.

### 3.4.2 VR Program

The VR program was the software that was responsible for handling everything in the virtual domain. This program generated the virtual environment that the user moved through and interacted with. It also determined how the user moved around and interacted with anything and everything in VR. The VSNs existed within this VR program.

#### 3.4.2.1 Virtual Environments

The virtual environments, often referred to as maps, are the surroundings and conditions in which the user operated in the virtual domain. These environments can often be summarized as the ground, the sky, sources of light, buildings, and objects such as benches tables and shelves that populated the virtual domain. The HMVR system used three maps that were designed for specific purposes that are described in Chapter 5. The VR program can switch through the maps as needed.

#### 3.4.2.2 VR User Interface

The VR user interface was designed to maximize the sense of immersion the user experienced, and to give the user the most efficient access to the various built in functions of the VR program. The user interface consisted of the user's ability to move through the virtual environment, virtual hands that are used to interact with the virtual environment, and the VR menu that is used to access various features of the VR program. These are discussed in detail in Chapter 5.

### 3.4.2.3 VSNs


*Figure 14: VSN*

The VR programs contained three VSNs that corresponded to the three sensor nodes in the WSN. A VSN in the VR program can be seen in Figure 14. These VSNs provided a way for data from the sensor nodes to be displayed in the virtual domain, as well as giving the user an interface to interact with the corresponding sensor node. The VSN consisted of a data backboard, several user inputs, and several visualization components. The various components of the VSNs could be picked up off the data backboard by the user and placed around the environment as needed.

### 3.4.2.4 Interactable Components

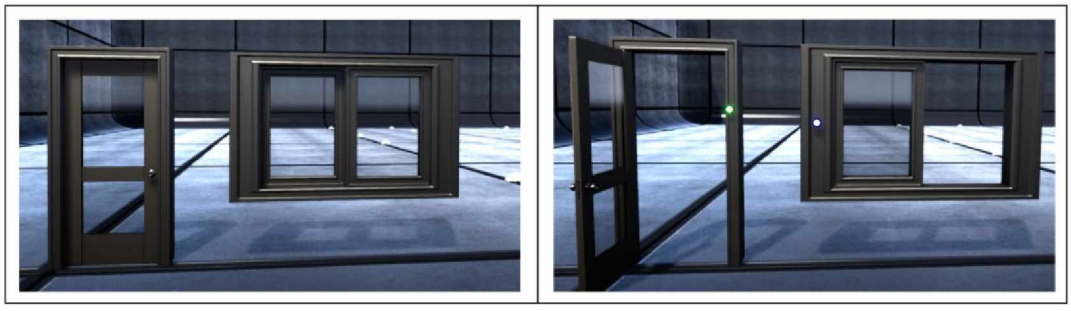

*Figure 15: Interactable Components*

The VR program also included interactable components in the virtual environments as seen in Figure 15. These components interacted with the sensor node data when a compatible

VSN visualization component was attached to them. For example, the switch status indicators could be attached to windows or doors. When the switch status was true, the window or door opened, and when the switch status was false, the window or door closed. This type of component was included to demonstrate how the VSN can feed data into other virtual components to modify their state and behavior in complex ways.

# Chapter 4 Hardware Implementation

The system developed in this thesis used a variety of different hardware components. These components ranged from the microcontroller in the WSN's nodes to the VR system. This chapter describes the physical hardware that was used, why the parts were chosen, and what the devices capabilities were.

## 4.1 WSN Components

As described in Chapter 3, the WSN consisted of three sensor nodes and one sink node. These nodes were powered by the USB devices that they were connected to. The sink node was powered by the laptop, and the sensor nodes could have been powered by a laptop, a USB wall outlet, or a portable USB power supply. This section discusses the hardware used to construct those devices.

### 4.1.1 Sensor Node

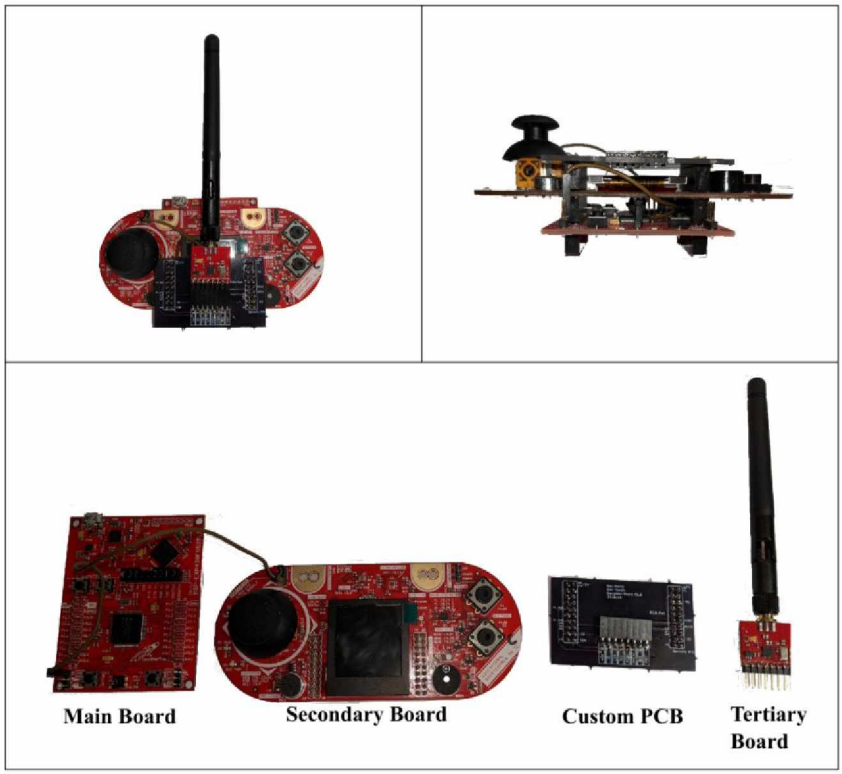


*Figure 16: Sensor Node, (Top Left) Top View, (Top Right) Side View, (Bottom) Exploded View*

The sensor nodes consisted of three off the shelf circuit boards and one custom built printed circuit board which can be seen in Figure 16. These circuit boards can be described as the main board, the interface board, the custom routing daughterboard, and the transceiver board. The main board (Texas Instruments MSP-EXP430F5529LP) held the microcontroller, voltage regulator, and USB interface. The interface board (Texas Instruments Educational Boosterpack MK II) held the sensors, buttons, an RGB LED, and a buzzer. The transceiver board (SparkFun Transceiver Breakout Board) held the RF transceiver and a connector for the antenna. The custom routing daughterboard was used to connect the transceiver board to the main and interface boards as well as route signals without the use of jumper wires. The main board, interface board, and the custom board were stacked vertically using a 4x10 pin bus called the boosterpack interface. This interface can be seen in Figure 16 (Top Right).

### 4.1.2 Sink Node

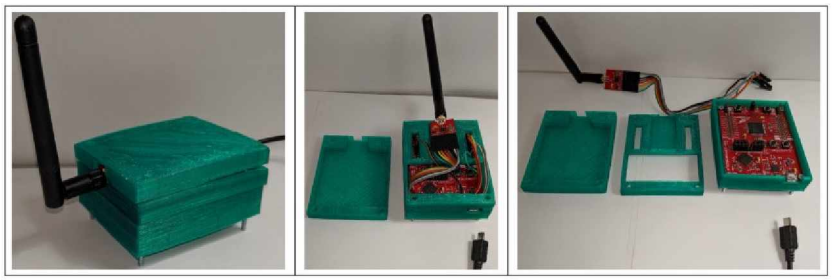

*Figure 17: Sink Node, (Left) Side View, (Middle) Open Case View, (Right) Exploded View*

The sink node was built using the same main board, transceiver board, and 2.4 GHz antenna as the sensor node. Instead of using a Custom Routing Daughterboard, jumper cables were used to connect the boards. A 3D printed case was built that protected the sink node from damage while keeping it compact and easy to place on a table next to the computer without getting in the way. A USB cable was used to connect the sink node to the computer to enable communication between the WSN and the VR program. Figure 17 shows several views of the Sink Node.

### 4.1.3 Texas Instruments MSP430F5529 LaunchPad Development Kit

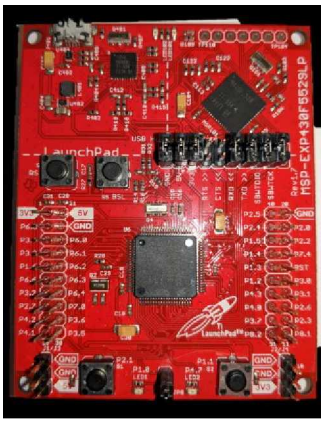

*Figure 18: TI MSP-EXP430F5529LP*

Both the sink node and sensor nodes used the Texas Instruments MSP430F5529 LaunchPad Development Kit for their main boards. This board was designed to give developers a platform to quickly construct a prototype system using the built in MSP430F5529 microcontroller. In addition to the microcontroller, a 40-pin (4x10) bus was used to break out pins from the microcontroller and share the 3.3 V, 5 V and GND rails. The 5 V rail was provided by the USB connection, while the 3.3 V rail was produced using a TPS62237 5 V to 3.3 V DC-DC converter. Four pushbuttons were included on the board: reset, bootstrap loader, and two programable buttons. Both the sensor node and sink node configurations made the bootstrap and reset buttons inaccessible, and the programmable buttons were only configured for debugging purposes. The top half of this board included an eZ-FET emulator that was used for programming the microcontroller, an integrated USB hub, the DC-DC converter mentioned earlier, and a USB connector that supplied the 5 V rail to the board while allowing USB communication. For the MSP430 microcontroller to communicate with a computer using UART in this configuration, the signals were sent to the eZ-FET lite Emulator MCU where it was converted into an emulated USB signal. This signal was passed to the USB Hub where it was transmitted over to the computer of the USB cable. In future iterations of this system, a simple

UART to USB converter like the FTDI FT232 would be much more power efficient. The block diagram of the board can be seen in Appendix A [49].

#### 4.1.3.1 Texas Instruments MSP430F5529 Microcontroller


*Figure 19: TI MSP430F5529LP Microcontroller*

Figure 19 shows the TI MSP430F5529 microcontroller, which is a very low power 25 MHz 16-bit MCU. A board was chosen with this microcontroller on it to produce a very power efficient system that was able to operate for a long period of time before running out of power. The system requirements demanded that the microcontroller was able to communicate with sensors over an I2C bus, the computer over a UART channel, and the RF transceiver over an SPI bus. The MSP430F5529 was capable of these tasks and more. When the microcontroller is properly configured and in a low power state, it is capable of operating while only drawing currents as low as 0.18 μA. The main modules of the MSP430F5529 microcontroller can be seen in Table 2 [50]. A functional block diagram along with active mode and low power mode supply current tables of the MSP430F5529LP can be seen in Appendix B.

*Table 2: MSP430F5529 Features*

| Feature | Value |
|---|---|
| Non-volatile memory (KB) | 128 |
| Ram (KB) | 8 |
| ADC | 12-bit SAR |
| ADC channels | 14 |
| GPIO pins | 63 |
| I2C | 2 |
| SPI | 4 |
| UART | 2 |
| Comparator channels | 12 |
| USB | Yes |
| Timers 16-bit | 4 |

#### 4.1.3.2 Texas Instruments TPS66237 Step-Down Converter

The Texas Instruments TPS66237 step-down converter was responsible for converting the 5 V rail coming from the USB connection into a 3.3 V rail that supplied power to most of the components on the WSN's nodes. This DC-DC converter had a typical quiescent current of 22 µA and was capable of outputting a peak current of up to 500 mA, and could operate with an efficiency of up to 94% [51].

#### 4.1.3.3 Texas Instruments TUSB2046 Full-Speed USB Hub

The HMVR system used the TUSB2046 Full-Speed USB Hub to communicate with the eZ-FET lite Emulator that was responsible for programing the MSP430 microcontroller. The USB Hub was also used as part of an interface that allowed communication between the microcontroller and the computer running the VR program. This device was fully compliant with the USB specifications as a full-speed hub and could draw up to 40 mA in normal operation. It is believed that this device and the eZ-FET lite Emulator are responsible for the main power consumption of sensor nodes. More power efficient components should be used in future designs to allow for a more efficient overall system [52].

### 4.1.4 Texas Instruments Educational Boosterpack MK II

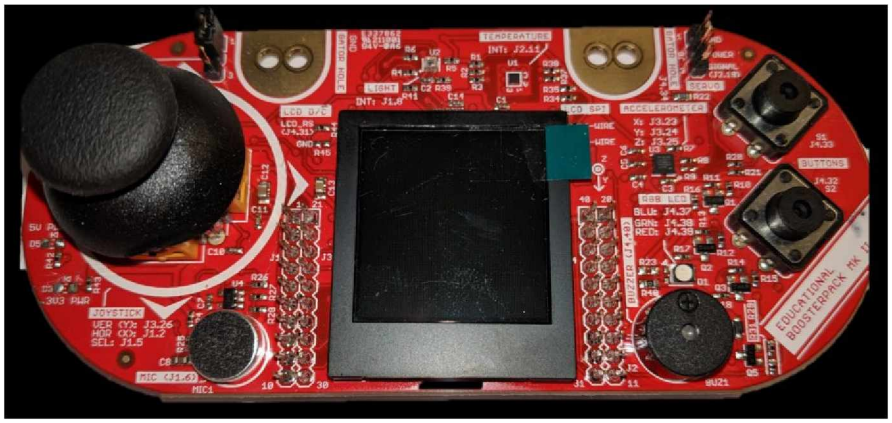

*Figure 20: TI Educational Boosterpack MK II*

The sensor node required several sensor and inputs, as well as the ability to respond to data from the VSNs. The interface board that was chosen for the sensor node was the Texas Instruments Educational Boosterpack MKII which can be seen in Figure 20. It was used because it had several sensors, user inputs, and data output options. This board used the same 40-pin bus that the main board used. The main components that were used on this board were the two push buttons, the 3-axis accelerometer, the temperature sensor, and the light sensor that allowed the sensor nodes to monitor their environment. The RGB LED and Buzzer were also used for responding to signals sent from the VSNs. This board also included a 2-axis analog joystick, a microphone, and a color TFT LCD screen which were not used. The LCD screen and its backlight were turned off to conserve power. An overview of the board's hardware can be seen in Appendix C.

#### 4.1.4.1 Kionix KXC9-2050 3-Axis Analog Accelerometer

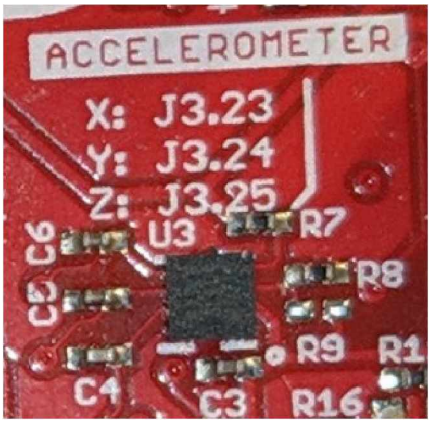


*Figure 21: Kionix KXC9-2050 3-Axis Analog Accelerometer*

The Kionix KXC9-2050 3-Axis Analog Accelerometer shown in Figure 21 is a low power accelerometer with a range of +/- 2 g. Its measurements are based on differential capacitance caused by acceleration of the internal sense element. The sensor was designed to minimize errors from process variation, temperature, and environmental stress. Three analog signals were output from the accelerometer representing the acceleration along the x, y, and z axes, where +2 g was represented by Vdd, and -2 g was represented as GND [53]. The MSP430 microcontroller measured the accelerometers outputs using three ADC inputs. A functional diagram, the mechanical, and electrical specifications of this device can be seen in Appendix D.

#### 4.1.4.2 Texas Instruments TMP006 Thermopile Sensor

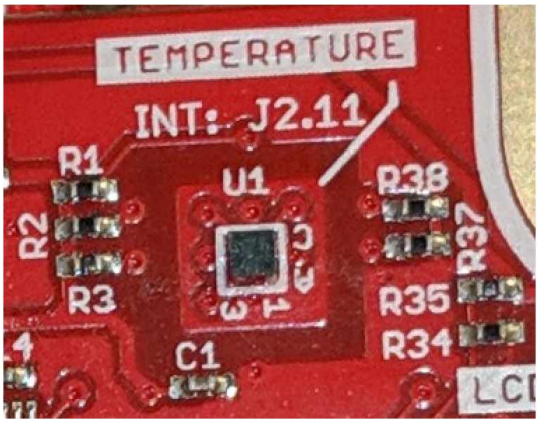


*Figure 22: TI TMP006 Thermopile Sensor*

The TMP006 Thermopile Sensor shown in Figure 22 was used to measure the ambient temperature at the sensor node. The sensor draws a low supply current of 240 µA and is capable of measuring temperatures between -40 °C and +125 °C (-40 °F to +257 °F). This chip was designed to measure the temperature of an object positioned in front of the sensor. However, in order to calculate the objects temperature, the chip needed to also accurately measure its ambient temperature. The sensor node was able to read this ambient temperature value using an I2C interface. An interrupt request pin was pulled low when a new temperature value was ready to be read. This interrupt pin alerted the MSP430 microcontroller to the newly available data. The sensor can be configured to provide measurements at different conversion rates as shown in Table 3 [54]. The functional block diagram and electrical characteristics of the thermopile can be viewed in Appendix E.

*Table 3: TI TMP006 Thermopile Sensor Conversion Rates [54]*

| CONVERSION RATE (conv/sec) | TOTAL NUMBER OF AVERAGED SAMPLES | AVERAGE $I_Q$ (µA) | PEAK-PEAK NOISE OF THE $T_{Object}$ RESULT (°C) |
|---|---|---|---|
| 4 | 1 | 240 | 0.5 |
| 2 | 2 | 240 | 0.35 |
| 1 | 4 | 240 | 0.25 (default) |
| 0.5 | 8 | 240 | 0.18 |
| 0.25 | 16 | 240 | 0.125 |

#### 4.1.4.3 Texas Instruments OPT3001 Ambient Light Sensor

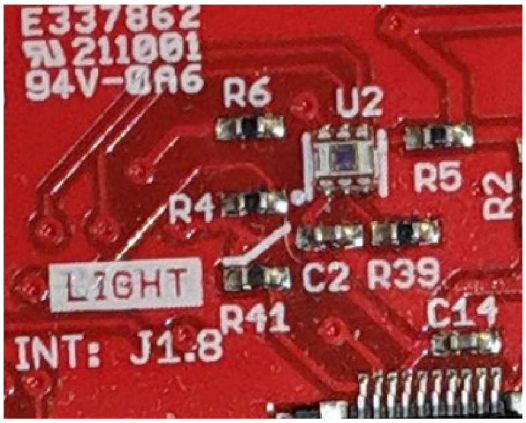


*Figure 23: TI OPT3001 Ambient Light Sensor*

The TI OPT3001 Ambient Light Sensor shown in Figure 23 was used to measure the ambient brightness at the sensor node. This sensor had a very low operating current of 13.8 µA and was able to measure brightness levels between 0.01 lux and 83000 lux. These brightness levels correspond with a moonless clear night sky with airglow [55] and direct sunlight [56]. It was designed to react to wavelengths of light similar to how a human eye would, as seen in Figure 24. This sensor had two selectable sample conversion times for measuring the ambient brightness: 100 ms and 800 ms which correspond to sample rates of 10 Hz and 1.25 Hz. The MSP430 microcontroller communicated with this sensor using the same I2C interface as the TMP006 Thermopile Sensor. Due to interrupt limitations in the configuration of the HMVR sensor nodes, the interrupt request pin for this sensor was tied to the same interrupt request pin as the thermopile. Both sensor interrupt request pins were open-drain with external pull up resistors resulting in the connected line being pulled low whenever either of the active low interrupt request pins were activated. It was left up to the microcontroller to determine which sensor had new data available. A functional block diagram and the electrical characteristics of this sensor can be seen in Appendix F [57].

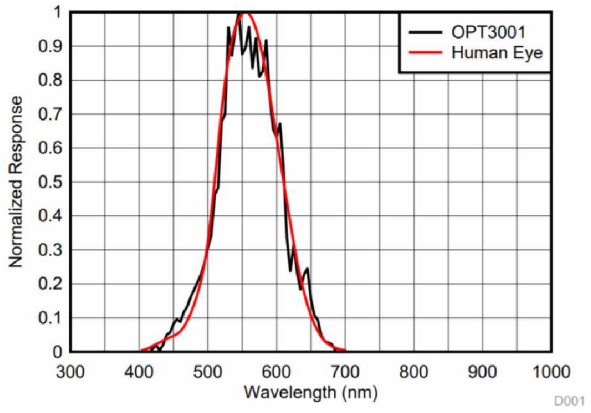


*Figure 24: TI OPT3001 Ambient Light Sensor Spectral Response vs Wavelength [57]*

#### 4.1.4.4 Cree CLV1A-FKB RGB Multicolor LED

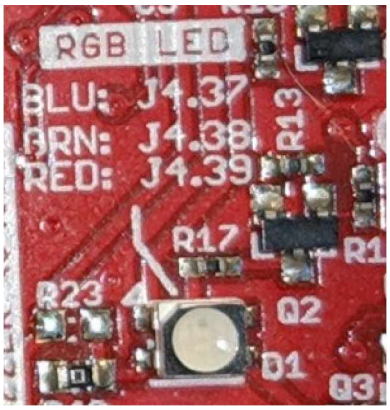


*Figure 25: Cree CLV1A-FKB RGB Multicolor LED*

The RGB LED used on the sensor node's interface board was the Cree CLV1A-FKB RGB Multicolor LED, which can be seen in Figure 25. This device was constructed using three LEDs placed in once package. The LEDs were driven individually and produced red, green, and blue lights. This component was used because it offered high intensity light output at a wide viewing angle and was driven by three PWM signals. This allowed for a low duty cycle PWM signal to produce sufficiently bright outputs. The RMVR sensor nodes were configured in such a way that light from the RGB LED was not detectable by the OPT3001 Ambient Light Sensor unless something was placed in front of the sensor node causing reflection. The typical electrical and optical characteristics of the Cree CLV1A-FKB RGB Multicolor LED can be seen in Appendix G [58].

#### 4.1.4.5 CUI CEM-1203(42) Piezo Buzzer

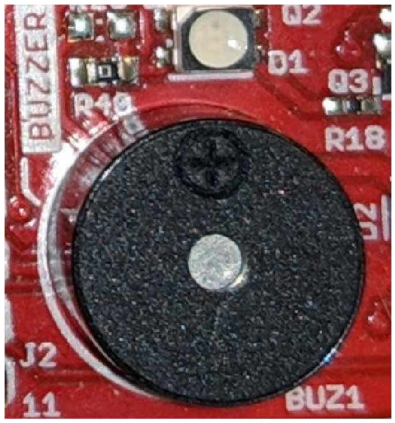


*Figure 26: CUI CEM-1203(42) Piezo Buzzer*

The CUI CEM-1203(42) Piezo Buzzer shown in Figure 26 was included on the board to enable audio output. The buzzer was rated for a 2048 Hz 3.5 V 50% duty cycle square wave input. A 95 dBA tone could be heard 10 cm from the speaker when the rated signal was applied. A mean current of 35 mA was drawn by the buzzer when the rated signal was applied. Due to the relatively high current draw of this device, it was only used to indicate specific and rare VSN input signals [59].

### 4.1.5 Custom Routing Daughterboard

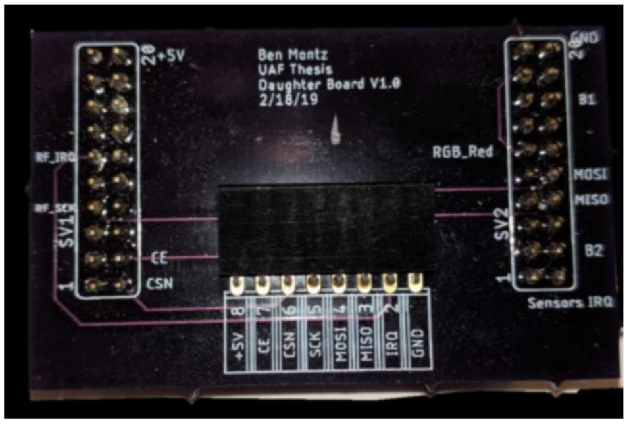


*Figure 27: Custom Routing Daughterboard*

The interface board was designed to connect with MSP432 based boards, and not the MSP430 based board used in the sensor node. This mismatch resulted in interface pins being

connected to microcontroller pins that did not have the required functionality. In order to provide important functions such as PWM signals to the RGB LED or use interrupts to detect button presses or sensor data, signals had to be rerouted from one pin to another on the 40-pin bus. The Custom Routing Daughterboard shown in Figure 27 was designed to reroute the signals as needed and to provide a header for the transceiver board. The pin connected to the red LED in the RGB LED was rerouted to a pin that corresponded to a pin on the MSP430, which allowed for the red, green, and blue LEDs to all be PWM driven by a single timer module. The pins that corresponded to the two input buttons on the interface board were rerouted so that they were connected to pins on the MSP430 that had interrupt capability. This was done to eliminate the need for polling the pins to detect a button press event, increasing the power efficiency of the sensor node. Finally, the pins corresponding to the light and temperature sensor interrupt request pins were tied together, as only one of these pins on the main board had interrupt capabilities. The schematic and layout for this board can be seen in Appendix H.

### 4.1.6 Sparkfun Transceiver Breakout Board

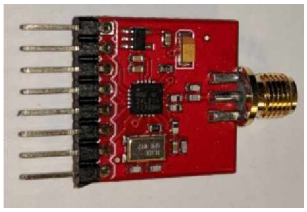

*Figure 28: Sparkfun Transceiver Breakout Board*

In order to communicate within the wireless network, an RF transceiver was needed. The Sparkfun Transceiver Breakout Board was chosen because it had both a powerful and capable transceiver, as well as a connector for an antenna. This breakout board is shown in Figure 28. In order to ensure a strong wireless connection between WSN nodes in the network, the version of the breakout board with an antenna connector was chosen to allow the use of a high gain antenna as described below.

#### 4.1.6.1 Nordic Semiconductor nRF24L01+ Single Chip 2.4 GHz Transceiver

The transceiver board used the Nordic Semiconductor nRF24L01+ Single Chip 2.4 GHz Transceiver. This transceiver was chosen as it can perform many important tasks without requiring input from the microcontroller and used relatively little power. It was capable of transmitting data with an air data rate of 2 Mbps at a transmission output power of 0 dBM. When powered down, this chip drew as low as 900 nA. The transceiver could draw up to 11.3 mA when transmitting at 0 dBm, or 13.5 mA when receiving data at 2 Mbps [60]. The functional block diagram and power consumption table of the nRF24L01+ transceiver can be found in Appendix I.

#### 4.1.6.2 2.4 GHz Dipole 2 dBi Antenna

*Figure 29: 2.4 GHz Duck Antenna*

The 2.4 GHz Dipole 2 dBi Antenna used in the WSN can be seen in Figure 29. As the name suggests, this antenna is a dipole antenna with a gain of 2 dBi and a frequency range of 2.4 to 2.5 GHz. [61].

## 4.2 Smart Home

The smart home setup used in this system, as described in previous sections, consisted of one Google Home Mini Google Assistant enabled smart speaker and three C by GE smart lights.

### 4.2.1 Google Home Mini Smart Speaker

*Figure 30: Google Smart Home Mini*

The Google Home Mini Google Assistant enabled smart speaker is a small and portable speaker that was chosen for this system because it had the Google Assistant built in. This smart speaker is shown in Figure 30. The Google Assistant is the Digital Assistant that this system uses to control the smart lights. The smart speaker needs access to a power outlet and the internet over a Wi-Fi connection in order to function.

### 4.2.2 C by GE Smart Lights


*Figure 31: C by GE Smart Lights*

The smart lights used in this smart home system were the C by GE smart lights, which can be seen in Figure 31. These lights have a peak brightness of 850 lumens and are controlled by the Google smart home using a Bluetooth connection [62]. The light could be set to any percentage of its maximum brightness using voice commands given to the Google Assistant.

## 4.3 Alienware 17 R4 Laptop Computer

*Figure 32: Alienware 17 R4 Laptop Computer*

The computer used for this system was the Alienware 17 R4 laptop as shown in Figure 32. This laptop was one of the several computers that Oculus recommended for use with the Oculus Rift VR system. The specifications of this laptop can be seen in Table 4.

*Table 4: Alienware 17 R4 Laptop Specifications*

| Specification | Value |
|---|---|
| Operating System | Windows 10 |
| Processor | Intel Core i7-7700HQ |
| Ram | 16 GB |
| Graphics Card | Nvidia GeForce GTX 1070 |
| Video Memory | 8 GB |
| Hard Drive 1 | 256 GB SSD |
| Hard Drive 2 | 1 TB HDD |
| USB Ports | 4 |

## 4.4 VR Hardware

There are several different VR systems that would have worked with this system. The Oculus Rift VR system was used as it offered excellent performance and immersion at a reasonable price. The Oculus Rift model used in this system is no longer available to consumers, but has been replaced with the Oculus Rift S, which provides a better experience without the

need for external sensors. The Oculus Rift S is expected to work with the HMVR system designed in this thesis without the need to make any modifications.

### 4.4.1 Oculus Rift HMD

*Figure 33: Oculus Rift HMD [63]*

The Oculus Rift HMD is a powerful and feature rich system. It was the most important device in the VR setup and can be seen in Figure 33. The Oculus Rift HMD used two OLED displays, one for each eye, that had a combined resolution of 2160×1200 with a refresh rate of 90 Hz. A pair of over the ear headphones were also built into the HMD to provide excellent audio output for a stronger immersive experience. The HMD used an accelerometer, a gyroscope, a magnetometer, as well as a constellation of built in IR LEDs that were tracked by the Oculus Sensors in order to monitor the precise location and orientation of the user wearing it. Figure *34* shows the built in IR LED constellation [64].

*Figure* 34*: Oculus Rift HMD IR LED Constellation [64]*

### 4.4.2 Oculus Touch Controllers

*Figure 35: Oculus Touch Controllers [63]*

A pair of Oculus Touch Controllers were used as the systems motion controllers. They can be seen in Figure 35. These motion controllers used an IR LED constellation system like that of the Oculus Rift HMD. The Touch controllers also included a 6-axis combinational gyroscope and accelerometer [65]. This combination of sensors and IR LEDs allowed for excellent hand tracking. The controllers used an assortment of capacitive touch sensors, buttons, joysticks, and triggers to precisely detect inputs from the user to allow a wide array of interactions in VR.

### 4.4.3 Oculus Sensors

*Figure 36: Oculus Sensor [63]*

The Oculus Sensor, as seen in Figure 36, used an IR camera to track the IR LED constellations built in to the Oculus Rift HMD and Oculus Touch Controllers in order to track and translate head and hand movements into the VR program. The HMVR system used two of these sensors to increase tracking precision and minimize tracking loss events.

# Chapter 5 System Software Design

The sensor node and sink node software design discussion in this chapter will cover how the MSP430's modules were configured to properly interact with the various components that were discussed in Chapter 4. The sensor node setup interface and the wireless network's communication protocol are also discussed. The design of the VR program is also described in this chapter. Its discussion will cover topics ranging from how the user moves around the virtual environment to how the visualization components interpret and present sensor node data to the user. The method used for modulating the WSN's activity level based on the visibility of the VSN is described in detail in Chapter 6.

## 5.1 WSN Design

The WSN in this system consisted of two main software designs, the sensor node and the sink node firmware. Each of these designs were custom made to perform their specific tasks. All the sensor nodes used the same software, but modified their functionality based on the sensor node ID they were given by the user through the setup interface shown in Section 5.1.1.9.

### 5.1.1 Sensor Node

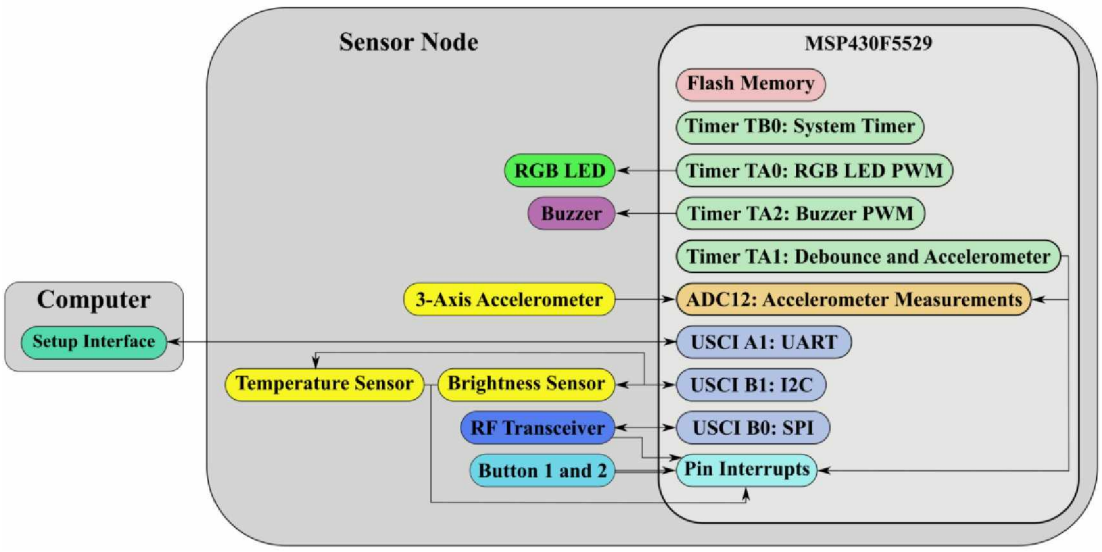


*Figure 37: Sensor Node Software Design Outline*

The sensor nodes made use of several hardware components as described in Chapter 4. In order to do so, the MSP430 microcontroller used many built in modules. Figure 37 shows an outline of the MSP430 modules that were used and how they interacted with the various components on the sensor node.

#### 5.1.1.1 Sensor Node MSP430

The sensor node's MSP430 was responsible for managing and carrying out all the operations on the sensor node. For the various modules and communication busses to function properly, the MSP430 clocks were set as shown in Table 5.

*Table 5: Sensor Node MSP430 Clock Frequencies*

| Clock | Frequency |
|---|---|
| MCLK | 16 MHz |
| SMCLK | 8 MHz |
| ACLK | 32768 Hz |

The sensor node had a set of five configurable variables to that were stored in Flash memory to enable recovery after a loss of power. These values were set using the setup interface discussed in section 5.1.1.9. These values were the sensor node ID, maximum brightness, minimum brightness, maximum temperature and minimum temperature. When the sensor node was powered up, the first thing it did was read the configurable values saved in the Flash Memory. If any of these values were not considered valid, all the values were considered invalid and default values were saved to flash. This set the values to a known state after programming the sensor nodes or when the Flash memory became corrupted. In order to meet the strict timing constraints of the MAC protocol used, great care was taken in the programing of the sensor nodes main program loop. Whenever the MSP430 microcontroller was not active it was sent into the "low power mode 1" to conserve power.

#### 5.1.1.2 Push Buttons

The sensor nodes used two button inputs as environmental inputs. The buttons used were momentary push to make switches that tied a microcontroller's input pin to low only for as long as the button was held down. The button inputs were intended as stand-ins for reed switches connected to windows or doors that would detect open and close events. Whenever a button was pressed, an internal variable representing the switch's state was toggled between true and false in order to replicate the behavior of the reed switches. The buttons that were used in the sensor node suffered from signal bouncing that caused erroneous false positive button press events. The microcontroller implemented a debounce system by using a timer module that disabled the button's interrupts for 30 ms after a button press was detected. Once a button press was detected, a software flag was set that indicate new data was available to send to the VSN.

#### 5.1.1.3 RGB LED

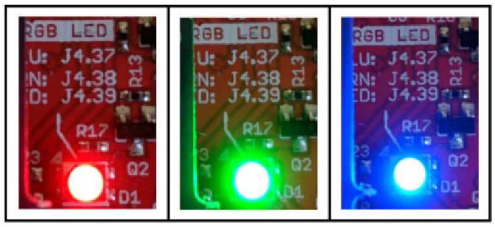


*Figure 38: RGB LED, (Left) Red LED, (Middle) Green LED, (Right) Blue LED*

The RGB LED was used to output the values of three Boolean variables that were toggled by signals coming from the VSN. The VSN had three color coded buttons (red, green, and blue). When any of these buttons were pressed, the corresponding Boolean value in the sensor node was toggled and the RGB LED color was changed accordingly as shown in Figure 38. The RGB LED was very bright, as discussed in Chapter 4, so the inputs were driven by three PWM signals that had a period of 12.2 ms and a duty cycle of 5% (610 µs). These values were determined experimentally to provide the desired visual output while minimizing power consumption.

#### 5.1.1.4 3-Axis Accelerometer

The accelerometer provided three analog inputs to the pins of the microcontroller with ranges of 0 V to 3.3 V representing -2 g to +2 g in the x, y, and z axes. In order to measure the analog voltage, the microcontroller used a built-in 12-bit analog to digital converter that sampled the values when instructed to do so. After the ADC measurements were collected, the values were calculated back to gs. The accelerometer data was constantly output on the ADC pins, so the instruction to measure the values did not rely on interrupt signals from the sensor. When the sensor node was in fast mode a timer was used to measure new values every 30 ms. However, when the sensor node was in slow mode new values were measured every time new light sensor data was ready. This enabled the timer to be disabled and reduces the sample time to 800 ms when in slow mode, reducing the power consumption of the sensor node. When new accelerometer values were read and processed a software flag was set indicating that new data was available to send to the VSN.

#### 5.1.1.5 Temperature Sensor

In order to measure the ambient temperature of the sensor node, the microcontroller used an I2C bus to communicate with the TMP006 Thermopile Sensor. The sensor was configured to measure the ambient temperature every 0.25 s or 4.0 s depending on the mode of the sensor node as explained in Section 5.1.1.7. The quarter second sample time was selected so that the virtual visualization component would be able to display a smooth temperature transition in the case of sudden temperature changes, like when a window is opened during the winter. The four-second sample time would produce noticeable discrete changes in the displayed temperature that would threaten the immersion of the system. The sensor alerted the microcontroller that new data was ready by pulling an interrupt request pin low. However, the interrupt pin was also connected to the ambient light sensor's interrupt pin. This meant that when the interrupt pin was held low, the

microcontroller communicated with the light sensor and then the temperature sensor to detect which sensor had requested the interrupt. If the temperature sensor had data ready to read, the microcontroller would collect the data from a register accessed using the I2C bus. Doing this would allow the sensor to release the interrupt pin. Once this was done, the measured value would be converted to degrees Celsius using the method shown in Appendix J. The temperature would then be converted to degrees Fahrenheit using Equation 1. When the new ambient temperature value was calculated, a software flag would be set that indicated new data was ready to be sent to the VSN.

$$°F = °C * 1.8 + 32 \quad (1)$$

#### 5.1.1.6 Ambient Light Sensor Setup

The ambient light sensor used the same I2C bus and the same interrupt pin as the temperature sensor. This meant that the same method was used to collect the ambient light as was used to collect the ambient temperature data. The ambient light sensor was configured so that the brightness would have a sample time of 800 ms when the sensor node was in slow mode, or 100 ms when the sensor node was in fast mode. When the new ambient brightness measurement was collected, the value was converted into lux using the method shown in Appendix K, taken from the OPT3001 datasheet. When a new brightness value was calculated, a software flag was set that indicated new data was ready for transmission to the VSN.

#### 5.1.1.7 Sensor Node Fast Mode and Slow Mode

The sensor nodes were set up in such a way that they could be placed in either a fast mode or a slow mode. These modes were used to modulate the activity of the WSN by reducing the rates at which the sensor nodes sampled new data values and transmitted data packets to the VSNs. The sensor node's mode is set by a value contained in the synchronization/data packets received from the sink node. Table 6 shows the sample times and rates of the sensors in both slow and fast modes.

*Table 6: Sensor Sample Times and Rates*

| | Slow Mode | | Fast Mode | |
|---|---|---|---|---|
| **Sensor** | **Sample Time (ms)** | **Sample Rate (Hz)** | **Sample Time (ms)** | **Sample Rate (Hz)** |
| 3-Axis Accelerometer | 800 | 1.25 | 30 | 33.33333333 |
| Ambient Light Sensor | 800 | 1.25 | 100 | 10 |
| Ambient Temperature Sensor | 4000 | 0.25 | 250 | 4 |

#### 5.1.1.8 Sensor Node RF Transceiver

The sensor nodes MSP430 microcontroller used an SPI bus to communicate with the RF transceiver described in Chapter 4. The RF transceiver was configured to transmit with a power of 0 dBm at an over the air data rate of 2 Mbps in order to provide the best radio link while minimizing the time that the transceiver was out of the power down mode. This was the most power demanding configuration but resulted in maximum network stability. These settings should be modified in future designs to meet the applications requirements. The RX and TX addresses of the transceivers were set in such a way that the sink node could behave as a "MultiCeiver", receiving data from multiple sensor nodes on the same frequency using multiple "data pipes" as shown in Figure 39. Communication using the transceiver was carried out using the protocol shown in Appendix M. The transceiver was only placed in an active mode when listening for a data/synchronization packet from the sink node, or if the sensor node had new

sensor data to transmit during one of its designated transmission windows. This functionality helped minimize the systems power consumption.

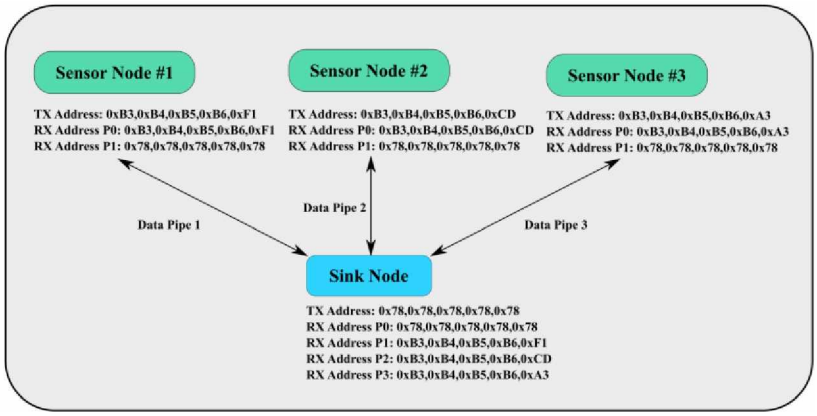


*Figure 39: MultiCeiver Transceiver Addressing Diagram*

### 5.1.1.9 Setup Interface


*Figure 40: Sensor Node Setup Interface*

The sensor node was configured using the setup interface shown in Figure 40. It was accessed by connecting the sensor nodes USB port to a computer. A terminal program such as Tera Term was used to connect to the sensor nodes COM port. The setup interface was toggled on or off by pressing the tilde key and used the MSP430s UART port configured as shown in Table 7.

*Table 7: Sensor Node UART Configuration*

| Setting | Value |
|---|---|
| Baud Rate | 115200 baud |
| Data Bits | 8 |
| Parity | None |
| Stop Bits | 1 |
| Flow Control | None |

This interface was used to set the configurable values discussed in previous sections. These values included the sensor node ID, the minimum and maximum brightness, and the minimum and maximum temperature values. The sensor node ID controlled various functional aspects of the system, from when to transmit data to the sink node, to what VSN should display the sensor node's data. The minimum and maximum temperature values indicated the user defined acceptable temperature limits for that sensor node. Any temperature measured beyond those values triggered an alarm in the VR program. The minimum and maximum brightness values determined what brightness measurement was considered 0% and 100% brightness. These values were used to calibrate the sensor node to detect various events. For example, a room with an open window would read a brightness above 0 lux when the lights were off. To detect if a light source of interest was on or off, the minimum brightness value would have to be set to the brightness value measured when the lights were off, and the maximum brightness value would be set to the brightness value measured when the lights were on. By calibrating the sensor node in this way, the VSN visualization component displaying the brightness data showed 0% when the light was off, and 100% when the light was on. The setup interface also displayed the current values from the sensor node's environmental inputs, the VSN's inputs, and some network details.

### 5.1.2 Sink Node

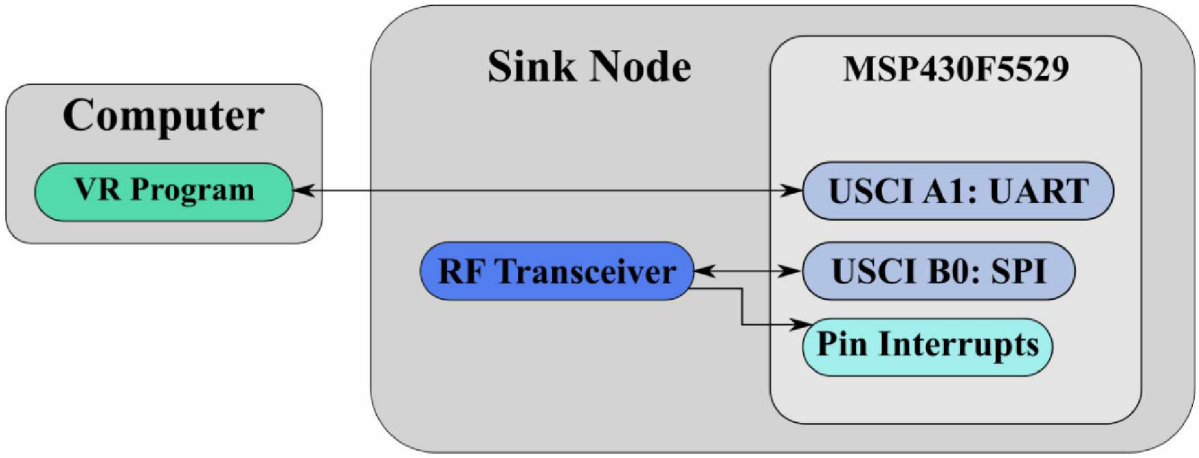


*Figure 41: Sink Node Software Design Outline*

Compared to the sensor node, the sink node was a simple but vital device. The software design outline of the Sink Node can be seen in Figure 41. As with the sensor node, the MSP430 microcontroller was responsible for controlling the behavior of the device. Unlike the sensor node, the sink node did not use any sensors, but was responsible for bridging communication between the VR program and the WSN. The sink node was also responsible for keeping the wireless network synchronized.

#### 5.1.2.1 Sink Node MSP430

The MSP430 microcontroller used in the sink node was set to use an MCLK frequency of 20 MHz which allowed the sink node to quickly and efficiently convert data packets coming from the sensor nodes into data packets using a protocol that the VR program can interpret and vice versa. The SMCLK and ACLK were set to 32768 Hz. These clock frequencies are also tabulated in Table 8.

*Table 8: Sink Node MSP430 Clock Frequencies*

| Clock | Frequency |
|---|---|
| MCLK | 20 MHz |
| SMCLK | 32768 Hz |
| ACLK | 32768 Hz |

#### 5.1.2.2 Communication with VR Program

The sink node was able to communicate with the VR program over a USB cable that formed a serial COM link between the computer and the MSP430s UART port. The UART port was configured in the same way as the sink node. This configuration can be seen in Table 7. The communication protocol used between the sink node and the VR program is described in Appendix L.

#### 5.1.2.3 Sink Node Transceiver Setup

The transceiver used in the sink node was configured the same way that was shown in section 5.1.1.8. However, the sink node was responsible for transmitting a data/synchronization packet every 200 ms as part of the communication protocol described in Appendix M. When the sink node's transceiver was not in transmit mode, it was in receive mode continuously listening for new data coming in from the sensor nodes.

## 5.2 Smart Home

The smart home system did not use any custom software, but instead used AI-based voice analysis algorithms designed by Google. The majority of the processing took place in the cloud, and therefore required a reliable internet connection.

## 5.3 VR Design

Everything that existed in the virtual domain in the HMVR system had to be designed and programmed in the VR program. This section discusses the various aspects of the software design of the VR program.

### 5.3.1 Virtual Environment

The virtual environments, or maps, were designed as explained in section 3.4.2.1. The environments provided a virtual area to move around in and objects to interact with. Three maps were designed and implemented in the HMVR system. Each one of these maps were designed with a specific goal in mind as described below.

#### 5.3.1.1 System Demonstration Map

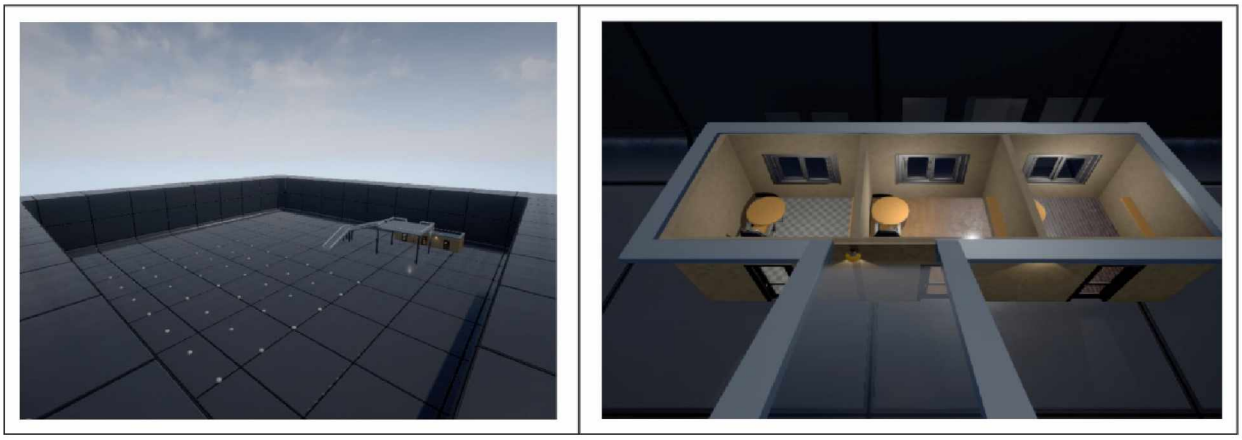

*Figure 42: System Demonstration Map, (Left) Elevated View, (Right) Three Room Observation Deck*

Figure 42 shows the most useful map included called the System Demonstration Map. This map was designed for the purpose of system testing and demonstration. It included many features that were useful for feature testing of the system. The map included three rooms and an observation deck that was used to monitor the rooms. This allowed for a demonstration of how the VSNs could translate the data coming from the WSN into a simulation of a room's environment. The map also demonstrated how the system could be used to monitor the environment of several rooms in a building simultaneously.

#### 5.3.1.2 UAF Map

*Figure 43: UAF Map, (Top Left) Google Earth View, (Top Right) VR Map Elevated View, (Bottom Left) Google Street View, (Bottom Right) VR Map Ground View [66]*

The second map included in the VR program was a 1:1 scale representation of a small section of the University of Alaska Fairbanks campus as seen in Figure 43. This map was designed to demonstrate that a virtual environment could be made to represent any environment that an application calls for from a university campus to a factory.

#### 5.3.1.3 Office Building Map

*Figure 44: Office Building Map*

The last map included in the VR program was the Office Building map, which can be seen in Figure 17. This map is included in the Unreal Engine that was used to build the VR program. The Office Building map was included to provide an example of the level of detail and realism that was achievable in a VR map.

### 5.3.2 Virtual Movement

Several methods of movement were built into the VR program to allow for a wide variety of functionality, immersion, and comfort. The first and most intuitive mode of movement is to physically walk around the physical environment. This translated the user's movement around their physical environment into identical movement in the VR program. This type of movement is limited by the amount of space that the user has available to move through. The second movement method that was implemented used the joysticks on the motion controllers. The joystick on the left motion controller was responsible for rotating the user, while the joystick on the right motion controller was responsible for translational movement. This method of movement was efficient and is used in many VR applications on the market but can lead to a form of motion sickness often called VR sickness. In order to prevent VR sickness, an option was included that obstructed the edges of the users view when moving as shown in Figure 45. These blinders have been experimentally shown to reduce VR sickness.

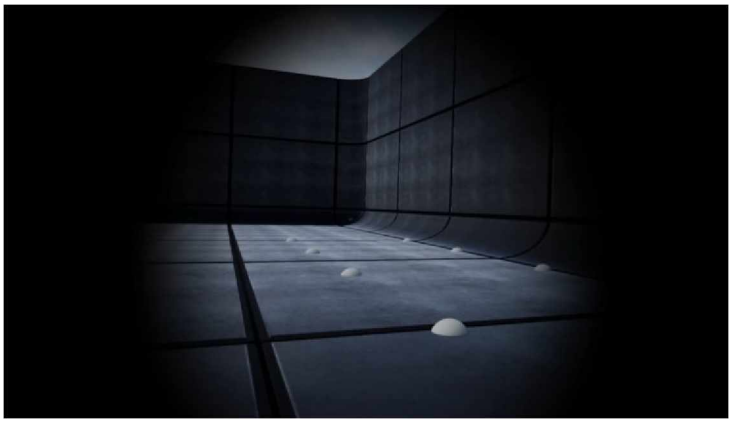

*Figure 45: VR Movement Blinders*

A third method of movement was also implemented that used teleportation to move the user from point A to point B quickly. The user began the process of teleportation by holding the right motion controllers' joystick forward, which produced an aiming display shown in Figure 46. The user then placed the teleportation destination marker on the desired location and pressed the trigger button on the right motion controller. The screen would then fade to black momentarily while the user was repositioned to the new location where the screen returned to normal. This method of movement enabled covering great distances in the virtual environment quickly without the risk of VR sickness.

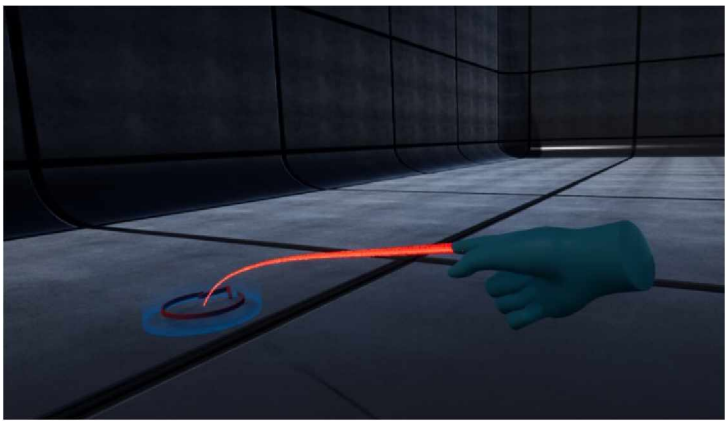

*Figure 46: VR Teleportation Aiming*

### 5.3.3 Virtual Hands

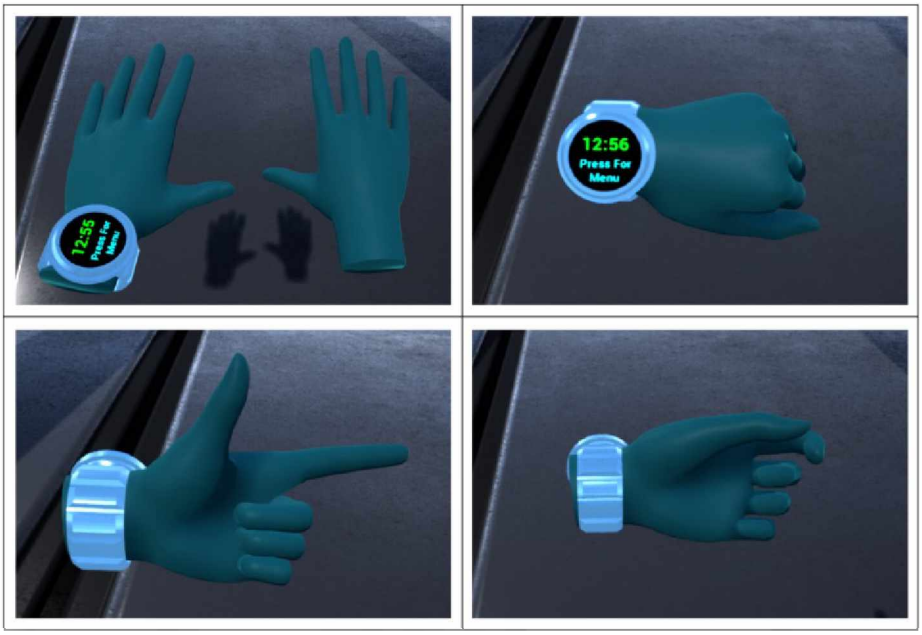


*Figure 47: Virtual Hands, (Top Left) Open Hands, (Top Right) Closed Fist, (Bottom Left) Pointing Hand, (Bottom Right) Partially Closed Hand*

Virtual hands were implemented that used the motion controllers tracking, capacitive touch sensors, and analog trigger data to closely replicate the position, orientation, and hand posture of the user. As Figure 47 shows, these virtual hands could be used for pointing, pushing buttons, and generally interacting with virtual objects. These virtual hands played a significant role in creating an immersive experience for the user.

### 5.3.4 VR Menu

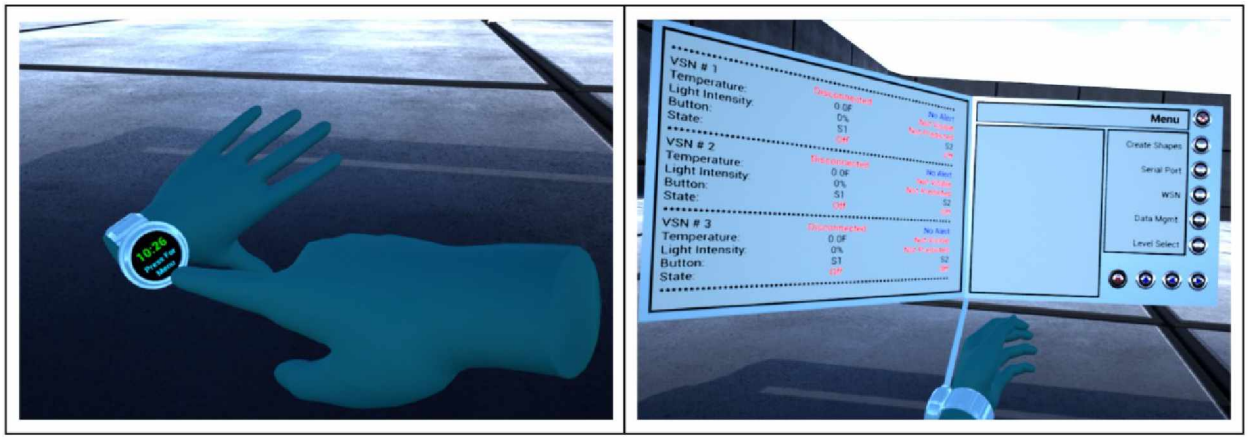


*Figure 48: VR Menu, (Left) Menu Activation, (Right) Extended Menu*

The VR menu shown in Figure 48 was included in order to enable access to various functions built into the VR program, such as accessing the serial port settings, connecting to the WSN, or changing the active map. The menu was designed in such a way that it was retracted into a virtual watch on the users left hand when not in use, but then extended from the user's watch when needed. This was used instead of a static menu placed in front of the user to enhance the immersion of the VR experience. The VR Menu consisted of two panels. The left panel displayed current data from the three sensor nodes. The right panel contained a frame that displayed context sensitive messages to the user like a console. The right panel also contained a dynamic menu system that used push buttons to navigate and select options with a top frame that displayed the name of the visible menu page.

### 5.3.5 VSNs

The VSNs were designed as a tool for displaying the environmental data coming from the sensor nodes placed throughout a home, as well as an interface to directly interact with the sensor nodes from within the VR program. Each of the three sensor nodes in the HMVR system corresponded to a specific VSN. This section discusses the various functions and features of the components that comprised a VSN.

#### 5.3.5.1 VSN Data Backboard

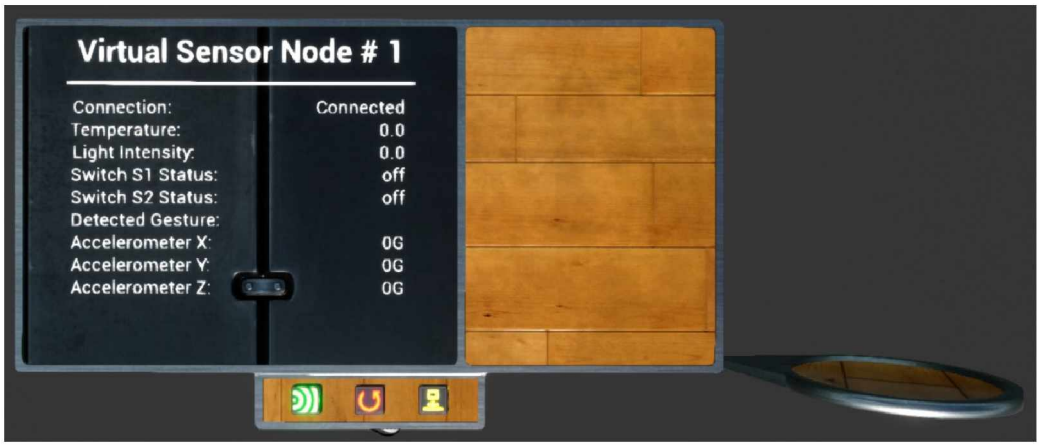


*Figure 49: Virtual Sensor Network Data Backboard*

The VSN's core component was the data backboard as seen in Figure 49. This object was used to display the text values of the most recent data received from the corresponding sensor node. The default locations of the various visualization and input components of the VSN were on the data backboard as shown in Figure 14. However, these components could be distributed throughout the virtual environment by the user as needed. The data backboard was designed as a display panel that showed the specific environmental values while visualization components were designed to process and display these values in a more tangible and meaningful way, as shown in Figure 50.


*Figure 50: VSN Backboard Displays*

The data backboard displayed several pieces of information including the ID of the sensor node that the VSN was displaying, the connection status of that sensor node, the current ambient temperature in Fahrenheit, the ambient light intensity as a percentage between the user defined maximum and minimum brightness values, the sensor node button statuses, the most recent gesture detected by the VSN, and the three accelerometer values in units of g.

#### 5.3.5.2 Visualization Components

*Figure 51: VSN Visualization Components*

The VSN contained several visualization components as shown in Figure 51. These components were designed to graphically display the sensor node's sensor data in a tangible and meaningful way in the virtual environment. These components displayed the various incoming data in many ways. A virtual lamp was used to simulate the ambient light, a virtual thermometer was used to denote the ambient temperature, two switch status indicators showed the state of the two sensor node buttons, and a 3D model of the sensor node simulated the orientation of the sensor node by processing the 3-axis accelerometer data. These visualization components are discussed in detail in the following sections. These components were designed with the intent of demonstrating that data coming from sensor nodes could be displayed in a variety of ways to produce meaningful and immersive VR experiences.

#### 5.3.5.2.1 Switch Status Indicator

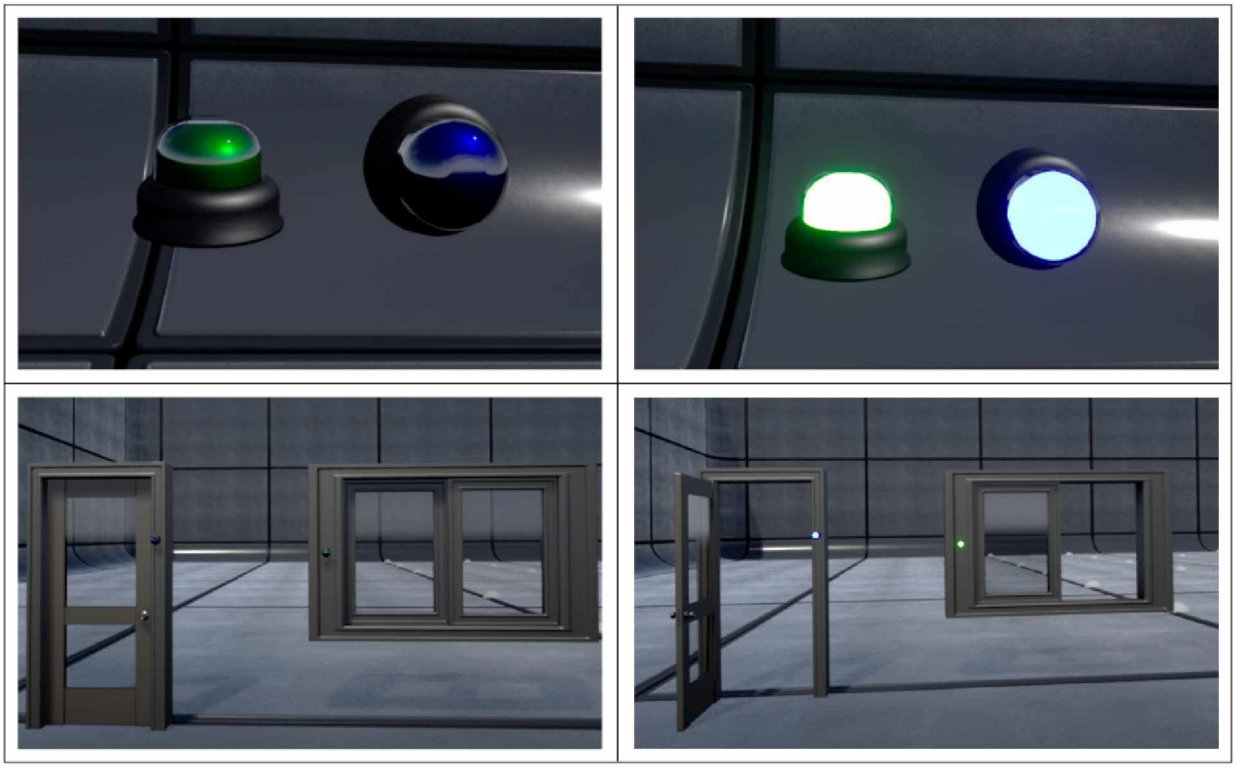

*Figure 52: Switch Status Indicator Functionality*
*(Top Left) Switch Status Indicators set to Off, (Top Right) Switch Status Indicators Set to On,*
*(Bottom Left) 'Off' Switch Status Indicators Attached to Interactive Door and Window,*
*(Bottom Right) 'On' Switch Status Indicators Attached to Interactive Door and Window*

A switch status indicator was designed in the form of a metallic puck that could self-illuminate on command encapsulated in glass attached to a metallic base. The VSN used two color-coded copies of these indicators to display the state of the sensor node's button inputs. The green switch status indicator corresponded with the 'button 1' state, while the blue switch status indicator corresponds with the 'button 2' state. When the switch status was set to 'on' the switch status indicators metallic puck became illuminated. However, when the switch status was set to 'off' the metallic puck was not illuminated. The user was able to grab the switch status indicators off the data backboard and attach it onto the interactable doors or windows discussed in Section 3.4.2.4. Once attached, the door or window's status mirrored the switch status indicators state. Figure 52 shows the Switch Status Indicator in operation.

#### 5.3.5.2.2 Sensor Node Orientation Indicator

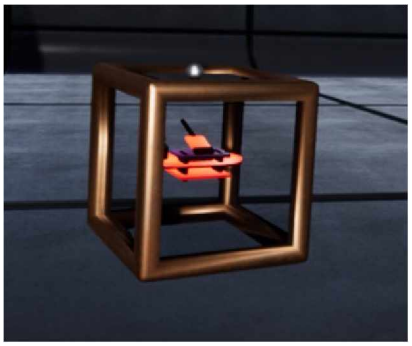

*Figure 53: Sensor Node Orientation Indicator*

The data coming from the sensor node's 3-axis accelerometer was visually represented using the sensor node orientation indicator, shown in Figure 53. This component processed the three acceleration values to calculate the orientation of the sensor node. The senor node's orientation was then simulated by a small-scale model of the sensor node enclosed in a glass and metallic frame with a small metallic sphere that was used to indicate the top face of the component. When the sensor node was rotated, the model of the sensor node mirrored the rotation inside the housing to match orientations.

#### 5.3.5.2.3 Virtual Thermometer

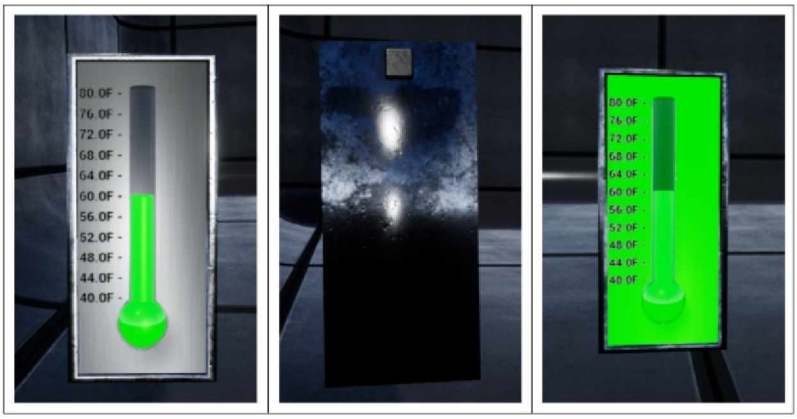


*Figure 54: Virtual Thermometer, (Left) Normal Status, (Center) Button on Back, (Right) Colored Backplate*

The ambient temperature value coming from the sensor node was displayed using a virtual thermometer shown in Figure 54. This component functioned in much the same way as a

classic bulb thermometer. A glass tube was filled with a material that raised or lowered to represent the ambient temperature. The thermometer had calibration markings that indicated the temperatures from the user defined minimum temperature to the maximum temperature in ten percent increments. The text on the thermometer could be difficult to read from a distance in VR so a toggle button was placed on the back of the virtual thermometer. Pressing this button configured the thermometer backplate to match the color of the material inside the glass tube as can be seen in Figure 54. The material inside the glass changed colors to indicate the temperature value in relationship to the minimum and maximum set temperatures. The thermometer's color was set to blue when the temperature was at the minimum temperature value and transitioned to green as the temperature approached the mid-point between the minimum and maximum values. As the temperature value approached the maximum temperature, the color displayed shifted from green to red. The color range of the virtual thermometer is shown in Figure 55.

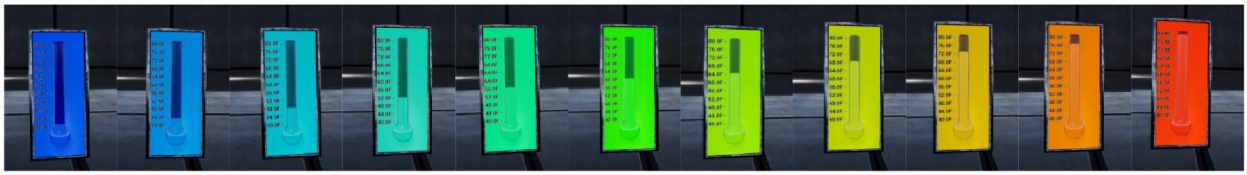

*Figure 55: Virtual Thermometer Color Range*

The ambient temperature in a room is a critical datapoint to monitor. An elevated temperature beyond a set limit could indicate a failed air conditioning unit or even a fire. A reduced temperature beyond the set limit could indicate a failure of the heating system or a broken window. It was imperative to alert the user when the sensor node's measured ambient temperature exceeded either the minimum or maximum temperatures. In order to do this, an alarm was triggered in VR and the area in front of the virtual thermometer was covered in snowfall or engulfed in flames as shown in Figure 56. This would give the user an immediate and blatant indication that the sensor node's room needed immediate attention.

*Figure 56: Virtual Thermometer Beyond Temperature Limits, (Left) Snow, (Right) Flames*

#### 5.3.5.2.4 Virtual Lamp

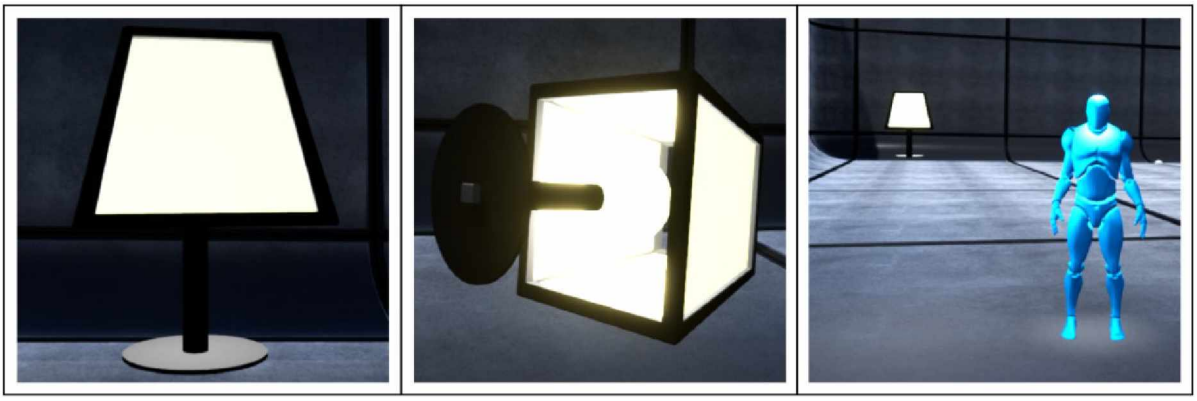

*Figure 57: Virtual Lamp, (Left) Normal View, (Center) Button, (Right) Presence Indication*

The ambient brightness value coming from the sensor node was displayed using the virtual lamp shown in Figure 57. The virtual lamp consisted of a frame with a button on it, a bulb and a lamp shade that self-illuminated, and a light source that illuminated the surrounding virtual environment. Figure 58 demonstrates the functional brightness range of the virtual lamp.

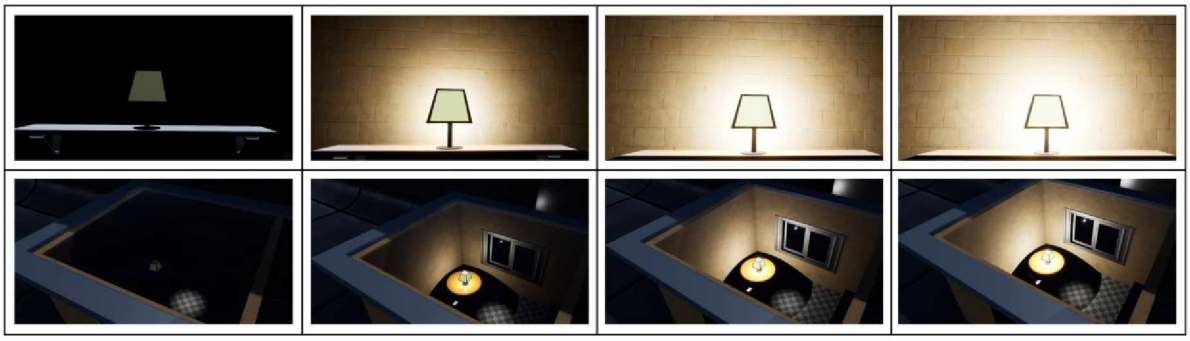

*Figure 58: Virtual Lamp Brightness Range*

When properly configured, the virtual lamp component that handled the ambient light data sent from the sensor node was able to determine if an object was present in the sensor node's room. This functionality was activated by pressing the button on the bottom of the lamp. When the presence detection feature was activated, a humanoid shape would appear next to the lamp when the brightness value dropped below 80% as shown in Figure 58 and Figure 59. The presence detection feature was able to detect when a person moved between the sensor node and the light source that it was calibrated to detect by detecting a drop in the brightness value as shown in Figure 60. The presence detection functionality was useful in an environment with constant unchanging lighting.

*Figure 59: Virtual Lamp Presence Indicator*

*Figure 60: Presence Detection, (Left) No Presence, (Right) Presence Detected*

### 5.3.5.3 Virtual Button Components

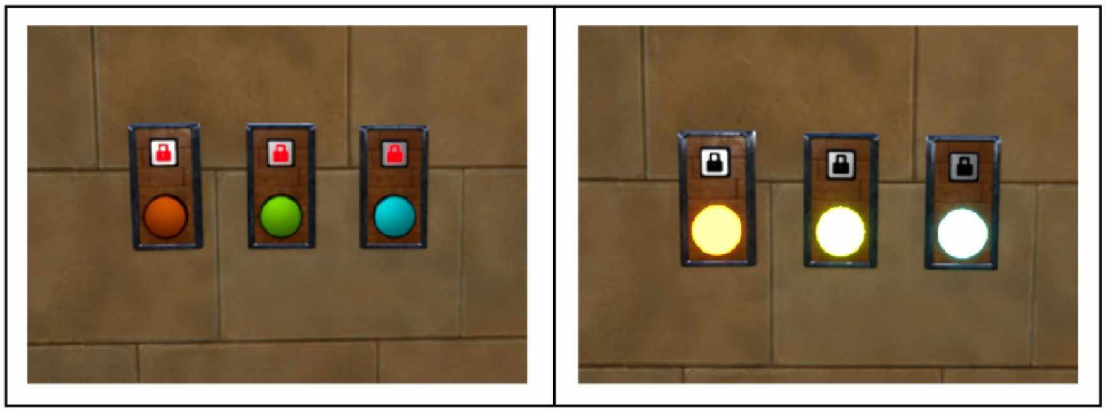

*Figure 61: Virtual Buttons*

The VSNs included three virtual button components that could be interacted with using the pointer finger of the virtual hands. The virtual button components had two buttons on their faces as can be seen in Figure 61. A lock button is included to prevent the buttons from being toggled when the component was being moved through the virtual environment. The main button was color coded to represent the buttons ID. Virtual button 1 was red and corresponded to the red element of the RGB LED on the sensor node. Virtual button 2 was green and corresponded to the green element of the RGB LED on the sensor node. Finally, virtual button 3 was blue and corresponded to the blue element of the RGB LED on the sensor node.

### 5.3.5.4 Virtual Gesture Detection


*Figure 62: Virtual Gesture Detection Interface*

The VSN's virtual gesture detection functionality was implemented to allow the HMVR system to detect when the user performed one of four specific gestures. When the functionality was activated, a glass box emerged from the face of the data backboard with two spheres representing the position of the user's virtual hands as displayed in Figure 62. This display was used to visualize the hand position data that was being analyzed by the program to detect gestures. When the VSN detected a gesture, a signal was sent to the corresponding sensor node that proceeded to emit a beep pattern that represented the gesture. The gesture detection functionality was included to demonstrate that the VR interface offered the user unique ways of interacting with WSNs. Table 9 shows the gestures that the system was designed to detect.

*Table 9: VSN Detectable Gestures*

| Gesture Name | Description | Sensor Node Beeps |
|---|---|---|
| Right Punch | Move right hand directly forward away from user. | 1 |
| Left Punch | Move left hand directly forward away from user. | 2 |
| Vertical Spread | Starting with both hands together, move one hand up and the other hand down. | 3 |
| Horizontal Spread | Starting with both hands together, move left hand to the left and right hand to the right. | 4 |

# Chapter 6 WSN Activity Modulation and Predictive Algorithm

The HMVR system implemented a novel algorithm for modulating the activity level of the WSN in order to conserve power, extend the lifespan of the WSN, and reduce the amount of RF signals in the environment that could interfere with other wireless systems. This method adjusted the rate at which the sensor nodes sampled environmental data and transmitted packets to the sink node based on the visibility of the corresponding VSN's visualization components in the VR environment.

## 6.1 WSN Activity Modulation

The sensor nodes were designed to operate in two modes: fast mode and slow mode, as discussed in Section 5.1.1.7. The rates that a sensor node collected new data was dictated by the mode that the sensor node was in. The sensor nodes only transmitted data packets when new values were ready, as explained in Appendix M. Therefore, changing the mode of a sensor node modulated the sensor sample rates and packet transmission rates. The sensor node used the slow mode instead of going into a power-down state where it stopped reading sensor values and transmitting data to the sink node so that the user was able to continuously monitor the sensor values using the menu-based sensor display described in Section 5.3.4. This also allowed the system to alert the user in the case of critical events such as the temperature going beyond the user defined limits. Table 10 shows the sensor sample rates in both the slow and fast modes.

*Table 10: Sensor Sample Rates*

| | Accelerometer Sample Rate (Hz) | Ambient Light Sensor Sample Rate (Hz) | Ambient Temperature Sensor Sample Rate (Hz) |
|---|---|---|---|
| Slow Mode | 1.25 | 1.25 | 0.25 |
| Fast Mode | 33.33 | 10 | 4 |

The system was designed so that a sensor node was set to fast mode when any of the corresponding VSN's visualization components were visible to the user in the VR program, otherwise the sensor node was set to slow mode. This configuration enabled a high-fidelity representation of the environmental data when the VSN was visible, while minimizing the power consumption and RF signals coming from the sensor node when the components were hidden from view. The system was then improved upon by including an optional algorithm that predicted if the VSN was going to be visible based on the movement of the user through the virtual environment. This improvement allowed the sensor nodes to transition into fast mode just before they entered the user's field of view. Doing so prevented the VSN from being visible during the mode transition, which could cause the visualization components to make sudden discrete changes to their output. These discrete changes could be jarring to the user and potentially break their immersion. The VSNs could be individually set to always be in fast mode, only check for the VSN's visibility, or check the VSN's visibility and predicted visibility. A flowchart for the WSN activity modulation algorithm can be seen in Figure 63.

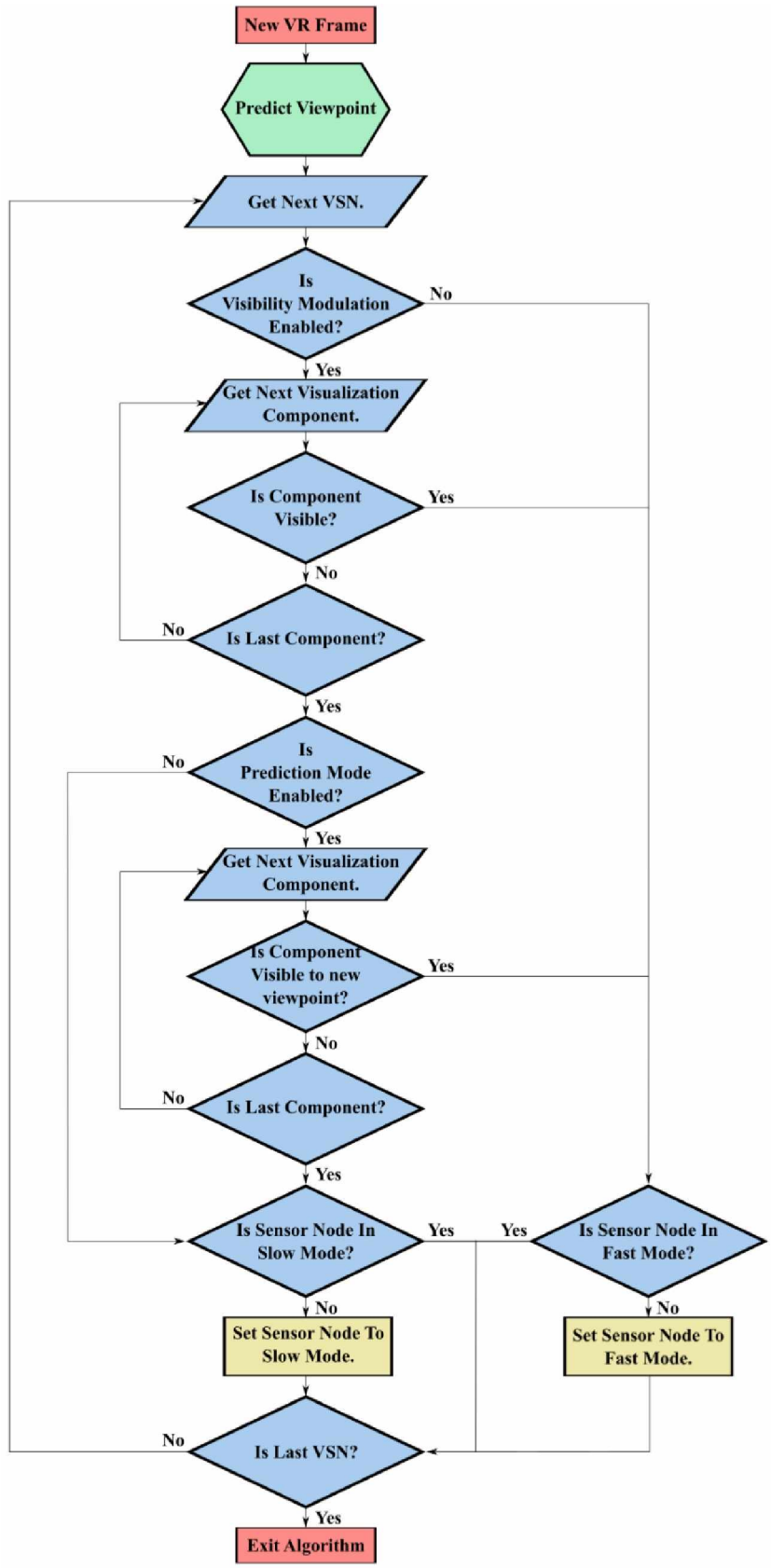


*Figure 63: WSN Activity Modulation Algorithm Outline*

### 6.1.1 Activity Mode Transition

When the WSN activity modulation algorithm determined that a sensor node needed to change modes, a message was immediately transmitted to the sink node using the communication protocol described in Appendix L. The next data/synchronization packet that the sink node transmitted, using the communication protocol shown in Appendix M, contained a value that instructed the sensor node to change activity modes. The sensor node then changed its activity mode as soon as the new packet was received.

## 6.2 Visibility Detection

The visibility status of the VSN was determined by the visibility statuses of the visualization components. If any one of the visualization components were deemed to be visible, the VSN would be considered visible and the algorithm would move on to testing the next VSN for visibility. The VR program was able to detect that a VSN was visible by using a value called "LastRenderTimeOnScreen" provided by the Unreal Engine that indicated how long it had been since the component was rendered onto the screen. This value was used to detect if any of the visualization components had been rendered on the screen in the two most recent frames delivered to the VR HMD. If any of the visualization components had been rendered within that time, the VSN was designated visible, otherwise the VSN was out of the user's line of sight.

## 6.3 Predicted Visibility Detection

As discussed above, the algorithm was also designed to predict if the VSNs were going to be visible based on the motion of the user. This was accomplished by first predicting the location and orientation of the user's viewpoint in the virtual environment based on the user's location, orientation, and average translational and rotational velocities. The algorithm then tested to see if any of the VSN's visualization components were visible to the user's predicted viewpoint. If any

visualization component was found to be visible to the predicted viewpoint, the VSN was predicted to be visible, otherwise the VSN was expected to not be visible.

### 6.3.1 Motion Prediction Algorithm

There are many algorithms that can predict the viewpoint of a user, from a simple linear extrapolation of the user's position and rotation based on instantaneous velocities, to predictions based on skeletal simulations. A simple algorithm was developed to predict the viewpoint of the user based on the average of the most recent translational and rotational velocities. This algorithm was designed with both simplicity of implementation and low computational cost in mind. Keeping the computational cost low was important as a new location prediction was made every VR frame. If the algorithm was too computationally intensive, the VR frame rate would suffer and risk breaking the immersion of the user. The viewpoint of the user was simplified to seven values: location vector(X,Y,Z), forward vector (φ,θ,ψ), and field of view (FOV) as shown in Figure 64. The location vector indicated the position in virtual world space, the forward vector indicated the direction that the viewpoint was viewing, given as roll (φ), pitch (θ), and yaw (ψ), and the FOV was the angle within which the viewpoint could detect the virtual environment.

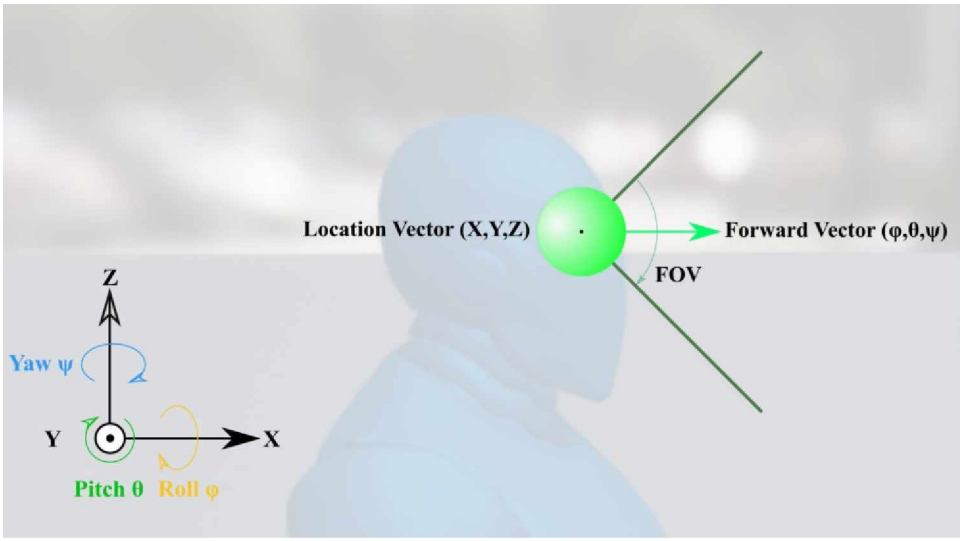


*Figure 64: Viewpoint Diagram*

The first step of the algorithm was finding an average of the forward vector and location velocities of the user's viewpoint over a set of the most recent frames. I determined experimentally that an average of 18 frames was enough to produce stable predictions. The velocities were calculated by finding the displacement between frames by subtracting the previous vector with the most recent vector as shown in Equation 2, then dividing that value by the time that elapsed since the previous frame, called "DeltaTime" as shown in Equation 3. The velocities were then averaged by adding the 18 most recently calculated velocities and dividing by 18 as shown in Equation 4. The averaged velocities were then multiplied by the prediction time to determine the predicted translational and rotational displacements as displayed in Equation 5. A prediction time of 400 ms was used as the prediction time in order to ensure the sensor node would be able to react before the VSN became visible, as the sink node only transmitted a packet every 200 ms. These values were then added to the user's current location and forward vectors, resulting in the predicted location and forward vector of the user as can be seen in Equation 6. Figure 65 shows a demonstration of the Motion Prediction Algorithm.

$$Displacement(x,y,z) = CurrentVector(x,y,z) - OldVector(x,y,z) \quad (2)$$

$$Velocity(x,y,z) = \frac{Displacement(x,y,z)}{DeltaTime} \quad (3)$$

$$AverageVelocity(x,y,z) = \frac{\sum_{n=1}^{18} Velocity_n(x,y,z)}{18} \quad (4)$$

$$PredictedDisplacement(x,y,z) = AverageVelocity(x,y,z) * PredictionTime \quad (5)$$

$$PredictedVector(x,y,z) = CurrentVector(x,y,z) + PredictedDisplacement(x,y,z) \quad (6)$$

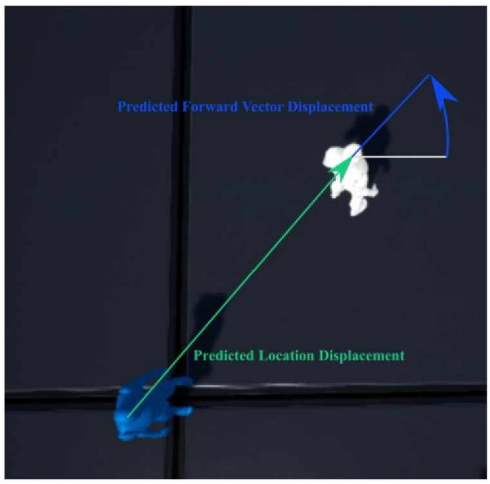


*Figure 65: Motion Prediction Algorithm Demonstration*

### 6.3.2 Predictive Field of View Selection

In order to account for visualization components that are just outside the field of view when the user is not moving, the field of view used for the predicted visibility detection algorithm was 30° wider than the field of view of the user. The user's field of view was 90°, while the predictive FOV was 120°. This ensured that sensor nodes corresponding to VSNs that could appear in the user's field of view with small movements were operating in fast mode.

### 6.3.3 Predicted VSN Visibility Detection

Detecting if the VSN's visualization components were going to be visible in the predicted viewpoint required a significantly more complicated process than was used to detect their visibility. There was no built-in function or value for this task. The algorithm discussed in this section detected if the visualization components that were within the FOV of the predicted viewpoint were going to be visible by checking if any points in a grid of points placed behind them were not obstructed by other virtual objects. If a single point behind a single visualization component was unobstructed from the predicted viewpoint the corresponding VSN was predicted to be visible. The point grid placed behind the visualization component was used in this algorithm so that it would not be excessively computationally intensive and would be most

likely to detect a visualization component that was partially obstructed by other virtual components.

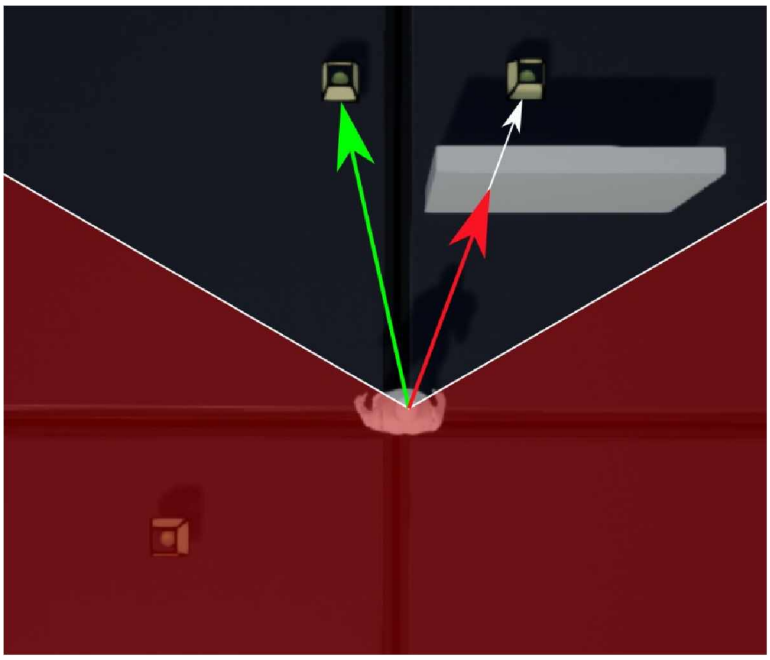

*Figure 66: Predicted Visibility Demonstration*

Figure 66 shows an example of the predicted visibility algorithm in action. The white humanoid figure represented the predicted viewpoint while the red area indicated the area that was outside of the prediction field of view. Three virtual lamps were shown in the figure to represent the visualization components. The virtual lamp in the bottom left was outside of the prediction FOV and was therefore not predicted to be visible. The virtual lamp in the top right was in the predicted FOV but the line of sight from the predicted viewpoint was obstructed by a wall and so it was also not predicted to be visible. The third virtual lamp that was in the top left was both inside the predicted FOV and its line of sight was not obstructed, so it was predicted to be visible. Figure 67 shows a simplified flowchart of the algorithm used to predict if a visualization component was visible.

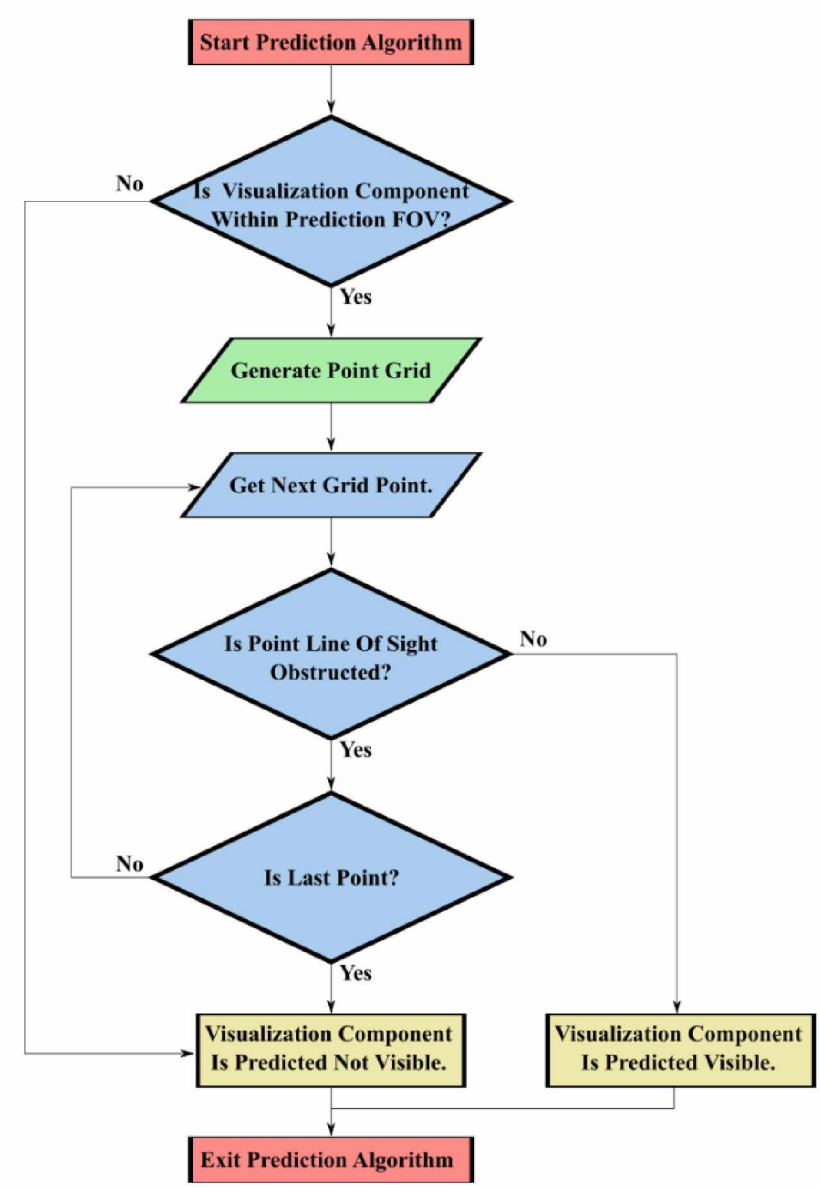


*Figure 67: Visualization Component Visibility Prediction Algorithm*

#### 6.3.3.1 Bounding Boxes

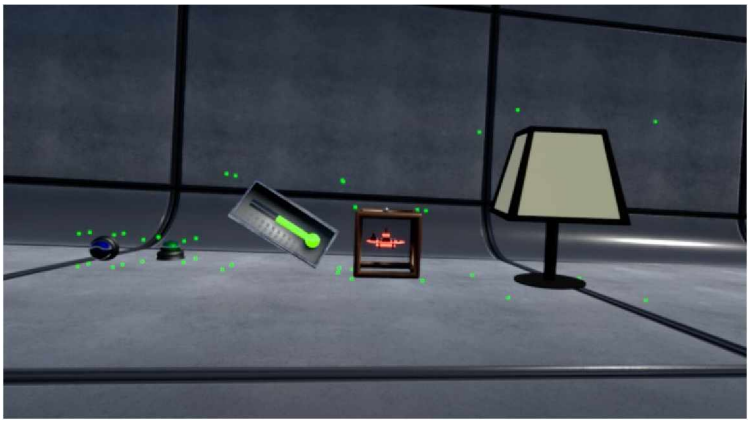

*Figure 68: Visualization of Bounding Box Corners*

The visualization components have built-in properties in the Unreal Engine that are called bounding boxes. These bounding boxes provide the positional extents of the components in the virtual world space. The corners of these bounding boxes were used to determine the size and location of the point grids used for the prediction algorithm. The bounding box corners of several visualization components are shown in Figure 68.

#### 6.3.3.2 Point Grid Generation

*Figure 69: Visualization of Point Grids*

The locations of points in the point grid that was set directly behind the visualization components was calculated for use in predicting the visibility of the components. The point grid was oriented normal to the forward vector of the predicted viewpoint. It was placed at a distance

that was equal to the distance from the predicted viewpoint to the bounding box corner that was farthest away. The size of the grid was determined so that it was just large enough to include all the bounding box corners from the perspective of the prediction viewpoint. The grid was made of 16 points in an evenly distributed 4x4 grid as shown in Figure 69.

#### 6.3.3.3 Line Trace

A function called "LineTraceSingleByChannel" that was built into the Unreal Engine played a significant role in this algorithm. This function scanned a linear path between two locations looking for the first component that intersected the line. The line trace was used to scan between the predicted viewpoint's location to the various points on the point grid. The algorithm was only interested in finding if any of the grid points were unobstructed, so if any of these traces detected that there was no object along the path, or that the object detected was the visualization component of interest, the VSN corresponding to the visualization component was predicted to soon be visible to the user.

# Chapter 7 Test Results

In addition to delivering a system that allowed for the management of the environments of multiple rooms in a building from VR, this thesis also developed a novel method for modulating the activity of a WSN based on what was or would be visible to the user in the virtual environment as described in Chapter 6. This chapter analyzes various aspects of the HMVR system's performance to demonstrate that it was well designed and that the WSN activity modulation algorithm significantly improved the efficiency of the system. This chapter also includes an analysis comparing the time it takes to report the brightness using the HMVR system vs requesting the lights brightness from the digital assistant to demonstrate the practicality of the system.

## 7.1 Activity Modulation Algorithm Analysis

The activity modulation algorithms described in Chapter 6 were tested to verify their functionality. This section discusses the results of these tests. The sensor nodes were found to successfully transition between fast and slow modes when visibility and predicted visibility states were changed. This was determined by monitoring both the values displayed on the sensor node's setup interface and the rate of transceiver interrupt requests when looking at and away from the VSNs.

### 7.1.1 Visibility Detection

The visibility of the VSNs were tested by monitoring the data on the left display of the menu shown in Figure 70. This display showed the visibility and predicted visibility states of the three VSNs while looking at and away from them. There was no discernable delay between when a VSN component came into view and when the menu verified that the VSN was visible. There did appear to be a very small delay between the VSN component leaving the user's view and the

menu verifying the VSN was not visible. This delay was caused by the Unreal Engine continuing to render the components momentarily after leaving the user's view. This delay had a negligible effect on the performance of the system.

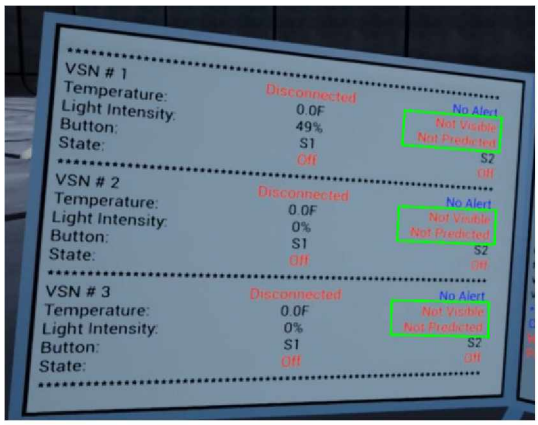


*Figure 70: Left Menu Display Showing VSN Visibility and Predicted Visibility States*

### 7.1.2 Motion Prediction

The motion prediction algorithm was verified using a test component that was designed specifically to show a representation of the users current and predicted viewpoint as seen in Figure 71. The blue shapes represented the user's current viewpoint, while the red shapes represented the predicted viewpoint. By monitoring this test component while turning and moving around the VR environment, it was verified that the motion prediction algorithm predicted viewpoints that would be visible if the user's movement continued uninterrupted. The averaging of velocities was determined to minimize the effects of short but sudden movements that would otherwise introduce large amounts of error into the viewpoint prediction.

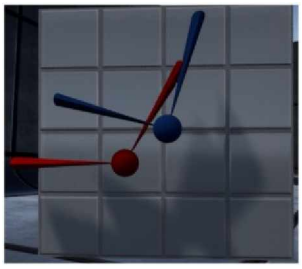

*Figure 71: Motion Prediction Test Component*

### 7.1.3 Visibility Prediction

By using the same testing method discussed in section 7.1.1, it was determined that the visibility prediction algorithm successfully predicted when a VSN's component was going to be visible before it became visible to the user. The VSNs were predicted to be visible early enough that the sensor nodes were able to transition from slow mode into fast mode before the VSN became visible. This verified that the visibility prediction algorithms were functioning as desired.

## 7.2 Sensor Node Power Consumption and Data Throughput

The power profiles of the HMVR system's sensor nodes were analyzed while in both fast and slow modes in this section. The current draw of the sensor node was measured using an oscilloscope and a sense resistor in series with a power supply to show to power profile of the sensor node in both modes. These profiles can be seen in Figure 72. This figure makes it very clear that the fast mode was much more active than the slow mode. It is also clear that the sensor node spent its time in either a low power mode, receiving RF data, or collecting sensor data and transmitting RF data.

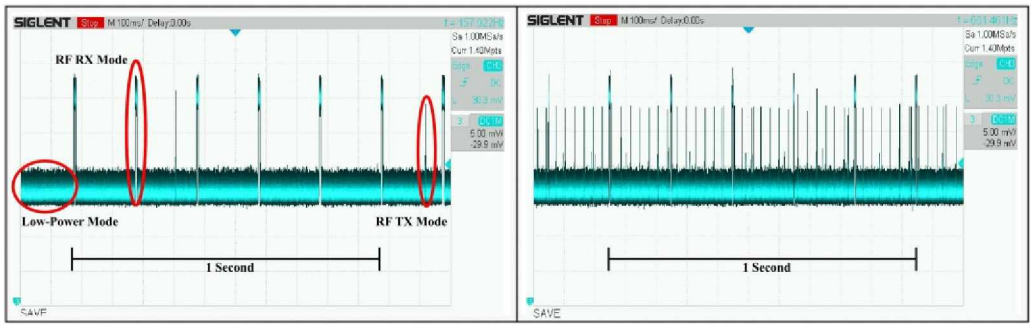


*Figure 72: Sensor Node Power Profile, (Left) Slow Mode, (Right) Fast Mode*

Unfortunately, the signals measured by the oscilloscope were too noisy and the other available test equipment did not have the resolution to measure the current accurately. As an alternative, current values from datasheets were used with precise timing measurements for the various system functions in order to analyze the power consumption of the system. Table 11

shows the typical currents of the components. To more accurately represent the expected measurements of fully custom-built sensor nodes, only the MSP430 microcontroller, transceiver, 3-axis accelerometer, ambient light sensor, thermopile, and voltage regulator were considered [50] [51] [53] [54] [57] [60].

*Table 11: Component Current Draw*

| Component: | Mode: | Current (µA) |
|---|---|---|
| MSP430 Microcontroller | Low Power Mode | 83 |
| | ADC Sampling | 233 |
| | Active Mode | 5700 |
| Transceiver | Standby | 26 |
| | TX Settling | 8000 |
| | Transmitting | 11300 |
| | RX Settling | 8900 |
| | Listening | 13500 |
| 3-Axis Accelerometer | Operating | 240 |
| Ambient Light Sensor | Operating | 3.7 |
| Thermopile | Operating | 240 |

Most of the component modes are self-explanatory, but the transceiver's TX Settling and RX Settling modes were the modes when the transceiver was switching to the TX or RX modes. The sensor node had several functions that required the components to be in specific modes. Table 12 gives a list of the system functions, their duration, expected current draw, and the corresponding modes of the MSP430 microcontroller and the Transceiver. The sensors were not listed in this table as they only use one mode. The system was in Low Power Mode when not performing any other functions.

*Table 12: System Functions*

| System Function | Time (µs) | Current (µA) | MSP430 | Transceiver |
|---|---|---|---|---|
| Low Power Mode | N/A | 592.70 | Low Power Mode | Standby |
| System Timer Interrupt | 12.56 | 6209.70 | Active Mode | Standby |
| Sample Accelerometer | 25.08 | 742.70 | ADC Sampling | Standby |
| Process Accelerometer Data | 83.65 | 6209.70 | Active Mode | Standby |
| Read Light Sensor Data | 259.34 | 6209.70 | Active Mode | Standby |
| Process Light Sensor Data | 48.85 | 6209.70 | Active Mode | Standby |
| Read Temperature Data | 549.64 | 6209.70 | Active Mode | Standby |
| Process Temperature Data | 110.57 | 6209.70 | Active Mode | Standby |
| Load TX Packet | 42.65 | 6209.70 | Active Mode | Standby |
| TX Settling | 130.00 | 8566.70 | Low Power Mode | TX Settling |
| RF Transmitting | 52.50 | 11866.70 | Low Power Mode | Transmitting |
| RX Settling | 130.00 | 9466.70 | Low Power Mode | RX Settling |
| RF Listening | 920.00 | 14066.70 | Low Power Mode | Listening |
| Read RX Packet | 81.23 | 6209.70 | Active Mode | Standby |
| Process RX Packet | 11.57 | 6209.70 | Active Mode | Standby |

The Low Power Mode system function was used when the sensor node had nothing to do and caused to system to minimize its power consumption. The sensor node spent most of its time in this state. System Timer Interrupt was a brief function that was used to check if data was ready to be transmitted. This function was performed at the start of every time-slot the belonged to the sensor node. RX Settling, RF Listening, Read RX Packet, and Process RX Packet were all functions used in the RF Receive system operation. These functions were responsible for setting the transceiver to RX mode, listening for data coming from the sink node, and then reading and processing that data. All three of the sensors have a sample or read function and a process function that are used to gather data from the sensors and then process that data into usable values. Once a sensor node value has been processed, the sensor node transmits the data using the Load TX Packet, TX Settling, and RF Transmitting functions that loads the data into the

transceiver, sets the transceiver into TX mode, and then finally transmits the data to the sink node.

The sensor node has five main operations that occur frequently. These include the system timer interrupt, receiving an RF packet, and sampling and transmitting accelerometer, light, and temperature data. Table 13 lists the system functions that occur during these operations. Figure 73 shows a diagram of the sensor node's operations and their corresponding system functions. The currents in these figures represent the current coming from the voltage regulator.

*Table 13: Sensor Node Operations*

| System Function | System Timer Interrupt | Accelerometer | Light | Temperature | RF Receive |
|---|---|---|---|---|---|
| System Timer Interrupt | ✓ | x | x | x | x |
| Sample Accelerometer | x | ✓ | x | x | x |
| Process Accelerometer Data | x | ✓ | x | x | x |
| Read Light Sensor Data | x | x | ✓ | x | x |
| Process Light Sensor Data | x | x | ✓ | x | x |
| Read Temperature Data | x | x | x | ✓ | x |
| Process Temperature Data | x | x | x | ✓ | x |
| Load TX Packet | x | ✓ | ✓ | ✓ | x |
| TX Settling | x | ✓ | ✓ | ✓ | x |
| RF Transmitting | x | ✓ | ✓ | ✓ | x |
| RX Settling | x | x | x | x | ✓ |
| RF Listening | x | x | x | x | ✓ |
| Read RX Packet | x | x | x | x | ✓ |
| Process RX Packet | x | x | x | x | ✓ |

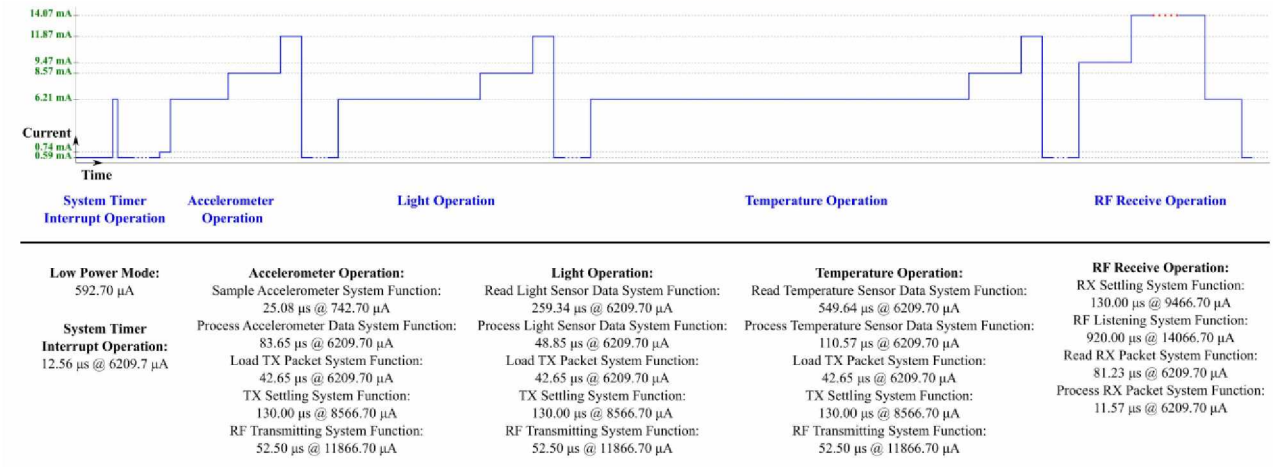


*Figure 73: Sensor Node Operation Diagrams*

By combining the information from Table 11, Table 12, and Table 13, the time and average current draw for each sensor node operation was calculated and shown in

Table 14. The currents given in this table represent the current drawn by the voltage regulator. These values were calculated using Equation 7 to account for the voltage difference and power losses from the 94% efficient voltage regulator, where P represents power and I represents current.

$$Efficiency = \frac{P_{useful}}{P_{input}} = \frac{3.3VI_{regulator}}{5.0VI_{system}} \quad (7)$$

*Table 14: Sensor Node Operation Values*

| Operation: | Time (µs) | Average Current (µA) |
|---|---|---|
| Low Power Mode | N/A | 416.151 |
| System Timer Interrupt | 12.56 | 4360.002 |
| RF Receive | 1142.80 | 9061.241 |
| Accelerometer | 333.88 | 5340.580 |
| Light | 533.34 | 5154.365 |
| Temperature | 885.36 | 4838.526 |

The frequency at which these operations occur are listed in Table 15. These frequencies change depending on if the sensor node is in the fast or slow mode.

*Table 15: Sensor Node Operation Frequencies*

| | Frequency (Hz) | |
|---|---|---|
| Operation: | Slow Mode | Fast Mode |
| System Timer Interrupt | 134.00 | 134.00 |
| RF Receive | 5.00 | 5.00 |
| Accelerometer | 1.25 | 33.33 |
| Light | 1.25 | 10.00 |
| Temperature | 0.25 | 4.00 |

Figure 74 shows the power profile of the system operations. Due to the noise and excessive ramp up and ramp down times in the test circuit, these measurements are only useful for approximating the current draw of the system.

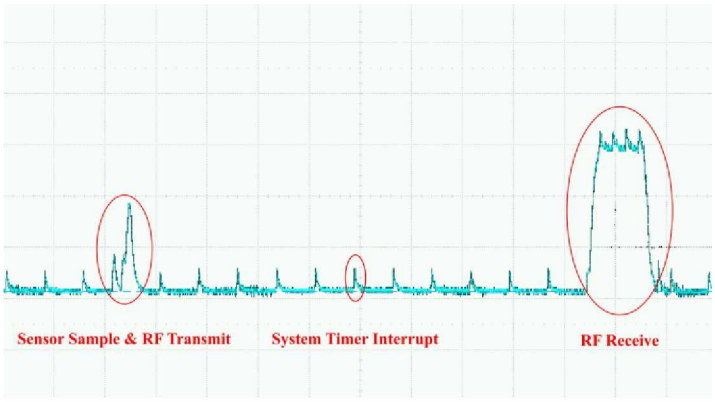


*Figure 74: Sensor Node Operations*

The average current of the sensor node was then calculated for both slow and fast modes using Equation 8. I represents current, f represents frequency, T represents operation time. The subscript "timer" represents the System Timer Interrupt operation variables, "RX" represents the RF Receive operation variables, "a" represents the accelerometer operation variables, "l" represents the Light operation variables, and "t" represents the Temperature operation variables. The expected lifespan of the sensor node was calculated using Equation 9, where $T_L$ represents the lifespan of the sensor node and $C_{Battery}$ represents the capacity of the battery. The data throughput of the sensor node was also calculated using Equation 10. $D_{Total}$ is the total data throughput of the system, while N represents the number of bytes in a TX or RX packet.

$$I_{Avg} = I_{timer}f_{timer}T_{timer} + I_{RX}f_{RX}T_{RX} + I_a f_a T_a + I_l f_l T_l + I_t f_t T_t + I_{LPM}(1 - f_{timer}T_{timer} - f_{RX}T_{RX} - f_a T_a - f_l T_l - f_t T_t) \quad (8)$$

$$T_L = \frac{C_{Battery}}{I_{Avg}} \quad (9)$$

$$D_{Total} = f_{RX} * N_{RX} + f_{TX} * N_{TX} \quad (10)$$

Table 16 shows the current draw, expected lifespan of the sensor node when using an ideal 2500 mAh battery, as well as the data throughput of the sensor node in both fast and slow modes.

*Table 16: Sensor Node Average Current and Lifespan*

| Mode: | Fast Mode | Slow Mode | Difference | % Difference |
|---|---|---|---|---|
| Average Current (µA) | 567.924 | 478.380 | -89.545 | -15.767 |
| Lifespan (days) | 183.417 | 217.749 | 34.332 | 18.718 |
| Data Throughput (bytes/s) | 256.665 | 33.750 | -222.915 | -86.851 |

This data clearly shows an improvement in power efficiency, lifespan, and data throughput between a sensor node in fast mode and one in slow mode. The reduced data throughput resulted in a reduction in RF congestion that could interfere with other RF systems. The improvements were relatively small for this case study design due to the relatively low data requirements of the system. The implemented design transmitted 5-byte data packets at a rate between 2.75 and 37.33 packets per second. However, the transceiver can transmit three 32-byte data packets in one transmission cycle, and a sensor node had 665 possible time-slots to transmit every second. A system with much higher data requirements designed to take full advantage of the transceiver's capabilities, the many time-slots, and the activity modulation functionality would result in a significant increase in the power and data throughput savings between slow and fast modes. The sensor nodes are not expected to spend their entire lifespan in one of the two modes. In order to calculate the current, lifespan, and data throughput for a sensor node that actively transitions between fast and slow modes, a variable A called the activity level was introduced. This value represented the percent of time that the sensor node was in the fast mode. Equations 9, 10, and 11 were used to calculate the sensor node's current consumption, lifespan, and data throughput over the full range of activity levels.

$$I_{Total} = I_{Fast\ Mode} * A + I_{Slow\ Mode} * (1 - A) \quad (11)$$

$$T_{L,Total} = T_{L,Fast\ Mode} * A + T_{L,Slow\ Mode} * (1 - A) \quad (12)$$

$$D_{Total} = f_{RX} * N_{RX} + f_{TX,Fast\ Mode} * N_{TX} * A + f_{TX,Slow\ Mode} * N_{TX} * (1 - A) \quad (13)$$

The results of these calculations can be seen in Figure 77, Figure 76, and Figure 77.

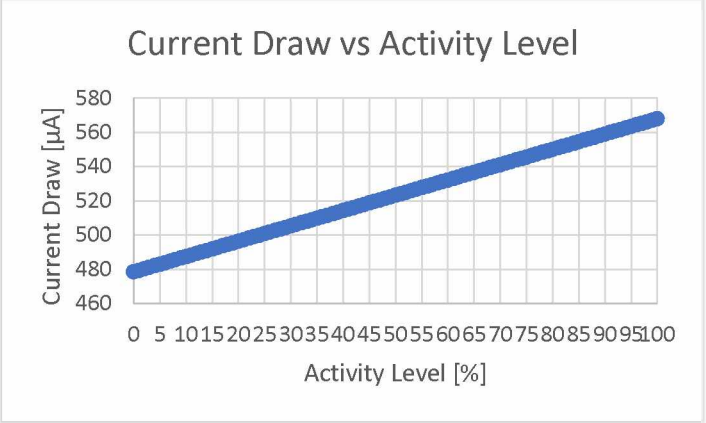


*Figure 75: Sensor Node Current Draw vs Activity Level*

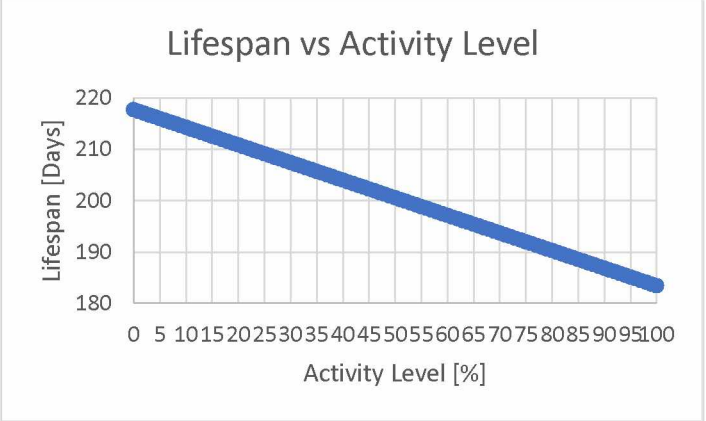


*Figure 76: Sensor Node Lifespan vs Activity Level*

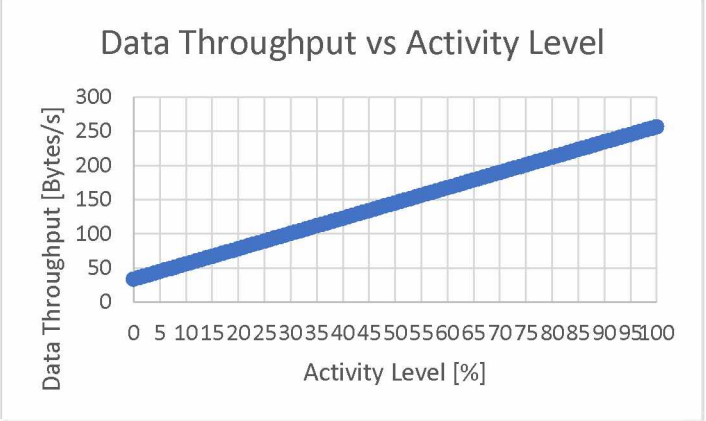


*Figure 77: Sensor Node Data Throughput vs Activity Level*

These figures show clearly that a system that fully takes advantage of the activity modulation algorithm described in this thesis can expect to see significant improvements to the power efficiency, lifespan, and data throughput of the system.

## 7.3 WSN Data Throughput

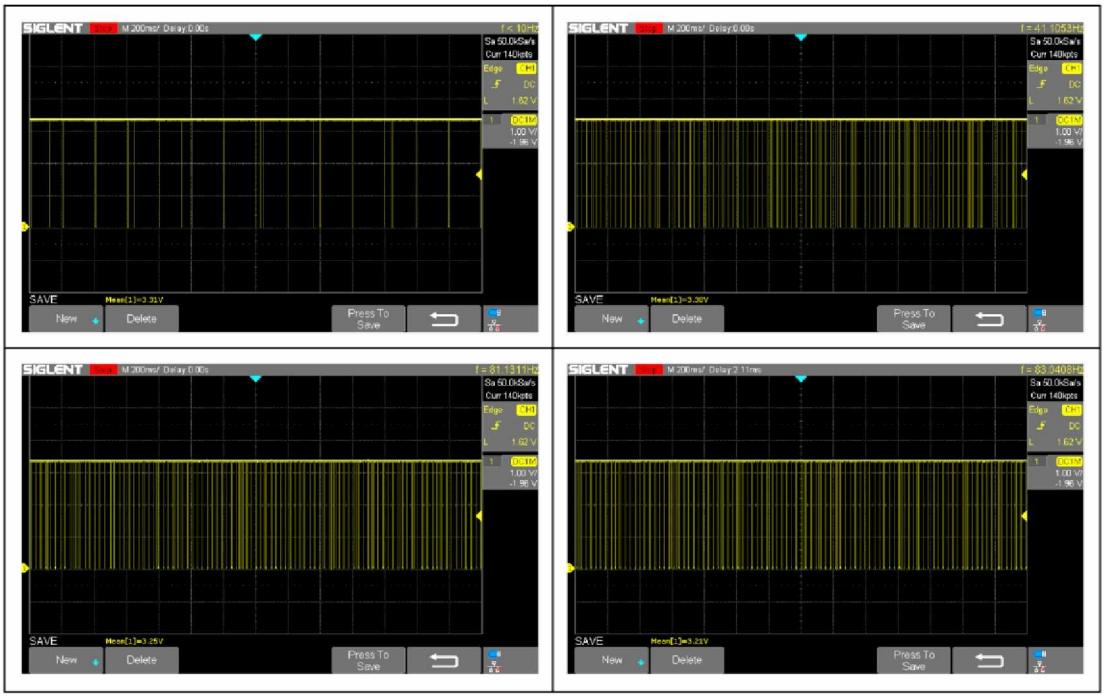

*Figure 78: WSN Data Throughput ,*
*(Top Left) 3 Sensor Nodes in Slow Mode, (Top Right) 2 Sensor Nodes in Slow Mode, 1 Sensor Node in Fast Mode, (Bottom Left) 1 Sensor Node in Slow Mode, 2 Sensor Nodes in Fast Mode, (Bottom Right) 3 Sensor Nodes in Fast Mode*

One of the main goals of this thesis was to design a system that reduced the data throughput of the WSN and minimized the RF noise in the home environment. As section 7.1 shows, the RF transmissions coming from an individual sensor node is dramatically affected by the activity modulation algorithm discussed in Chapter 6. Figure 78 shows the network data throughput of the WSN by monitoring the IRQ pin of the transceiver. This pin was pulled low every time the sink node's transceiver transmitted or received an RF data packet. Figure 78 shows the data throughput of the HMVR's WSN while connected to the three sensor nodes that were placed in the various activity mode configurations. These configurations were: 3 sensor

nodes in fast mode, 2 sensor nodes in fast mode and 1 in slow mode, 1 sensor node in fast mode and 2 in slow mode, and finally 3 sensor nodes in slow mode. The figure shows how significantly the activity modulation affected the RF activity of the WSN.

## 7.4 Digital Assistant vs VSN Environmental Query

In the HMVR system, two subsystems could query environmental factors such as the ambient light. The sensor nodes provided these values to the user in VR, while the smart home system delivered the value using the Google Assistant. The smart home system was included in the system to give the user the ability to adjust the environment, but it also had the ability to query the status of the smart lights. This section demonstrates that the smart home system did not make the sensor nodes redundant.

The Google Assistant was able to query the status of the smart lights in the smart home. However, this feature was limited in the ways that the data could be presented to the user. The responses to the query were limited to "on" or "off" values. An experiment was conducted where a user queried the Google Assistant for the status of the lights in order to monitor the time it took for the assistant to respond with the statuses. This experiment was conducted in three configurations with 1, 2, and 3 smart lights connected to the system. The query and response interaction went as follows:

User: "Hey Google, are my lights on?"

Google Assistant:

(one light setup) "The sensor node 1 light is on."

(two light setup) "The sensor node 1 light and the sensor node 2 lights are on."

(three light setup) "The sensor node 1 light, the sensor node 2 light, and the sensor node 3 lights are on."

The average response times of ten tests are shown in Table 17. The response times of the Google Assistant are lengthy and were determined mostly by the number of connected smart lights, their assigned names, and their statuses.

*Table 17: Google Assistant Smart Light Query Response Times*

| Number of Smart Lights | Response Time (s) |
|---|---|
| One Light | 3.822 |
| Two Lights | 5.214 |
| Three Lights | 6.960 |

The sensor nodes have many more capabilities than the smart home as described in the previous chapters. When the VSN displays the brightness values coming from the sensor node, it shows the relative brightness of the smart lights instead of just indicating if they are on or off. This is already a significant improvement over the Google Assistant's capabilities. An experiment was also carried out to measure the time it took for sensor node data to appear on the VSN's visualization components. This experiment was repeated ten times for configurations where 1, 2, and 3 sensor nodes were connected to the WSN. The average response times of these experiments are shown in Table 18.

*Table 18: VSN Response Times*

| Number of Sensor Nodes | Response Time (s) |
|---|---|
| One Sensor Node | 0.135 |
| Two Sensor Nodes | 0.185 |
| Three Sensor Nodes | 0.225 |

From this table it is clear that the sensor nodes can transmit instantaneous environmental data to the user much quicker than the Google Assistant is capable. It is also important to recognize that querying the Google Assistant for the environmental status provides one set of

values, while the sensor nodes provided continuous environmental data to the VSN. This made it very clear that the smart home system integration does not make the sensor nodes redundant.

# Chapter 8 Conclusion and Future Work

This thesis demonstrated that a system utilizing VR, smart home, and WSN technologies can be efficient and useful in a variety of applications. The HMVR system developed in this paper was inexpensive and made with mostly off the shelf components. This made it affordable and accessible for further research and development. This system successfully monitored environmental conditions in multiple rooms and visualized the collected data in in a meaningful and tangible way inside a virtual environment, while allowing the user to control the physical environmental conditions with voice commands given to the digital assistant. The system also allowed the user to remotely interact with WSN's sensor nodes from inside the VR environment by way of the VSNs. The VR program demonstrated that environmental data being collected by the WSN could be used to both alter the virtual environment and be further processed to sense events of interest. The data processing used in this system was relatively simple but was intended as a proof of concept and as an example to demonstrate how more complex systems could be designed to produce much more sophisticated results in a virtual environment by combining and processing several sensor inputs synergistically. This thesis revealed that while the system may at first appear to be an alternative solution to using a smart home app or digital assistant to monitor environmental factors, it is significantly more efficient and effective at presenting environmental data to the user as discussed in Section 7.3. High-fidelity representations of the various environments monitored by the WSN were produced in VR while maintaining a relatively low power profile by modulating the activity of the WSN's sensor nodes. This activity modulation was dependent on the visibility or predicted visibility of the VSNs in the virtual environment. The use of activity modulation not only resulted in a significant reduction of RF signals that could interfere with other wireless systems, but also reduced the power consumption of the sensor nodes. This would increase the life span of the WSN if the sensor nodes were

powered using USB power banks or batteries. This paper also showed that the HMVR system did not take full advantage of the activity modulation algorithm, and that a system that had larger difference of RF transmission rates between fast and slow mode could expect to see significant improvements in power efficiency.

Several improvements could be made to the future iteration of the HMVR system that would drastically improve its functionality and value for use in future applications. Significant improvements could be gained by constructing custom circuit boards for the sensor and sink nodes that used more efficient components and did not have unnecessary parts. By converting the WSN topology from single-hop star to a two-tier hierarchical cluster, it would be possible to place specialized sensor nodes closer to important environmental factors as shown in Figure 79. This improved system would gather more accurate environmental data [4].

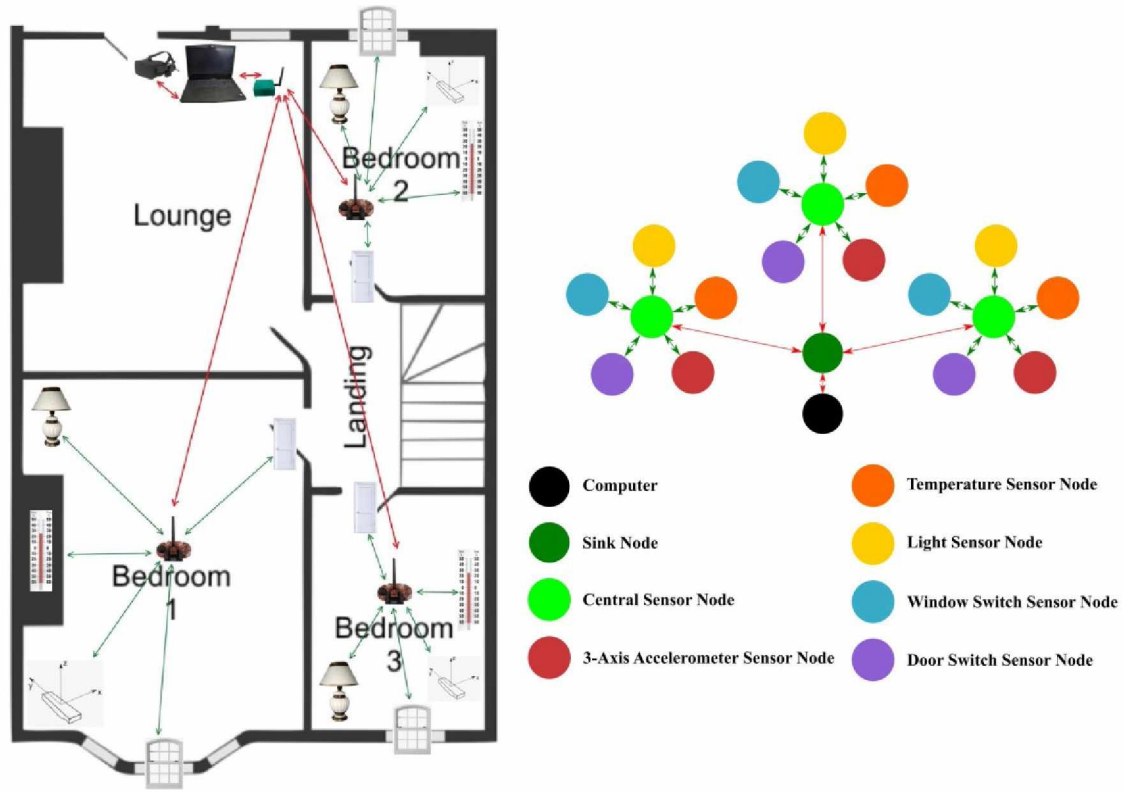


*Figure 79: HMVR System Using Two-Tier Hierarchical Cluster Topology, (Left) Floor Layout, (Right) Network Topology Diagram*

By using a Wi-Fi or Cellular Data antenna to connect the sink node to a server, and having the computer connect to that sever over the Internet, a user would be able to use the system at great distances from where the WSN is located as shown in Figure 80. This new setup would also allow standalone and mobile VR systems that do not have UART capabilities to use the system.

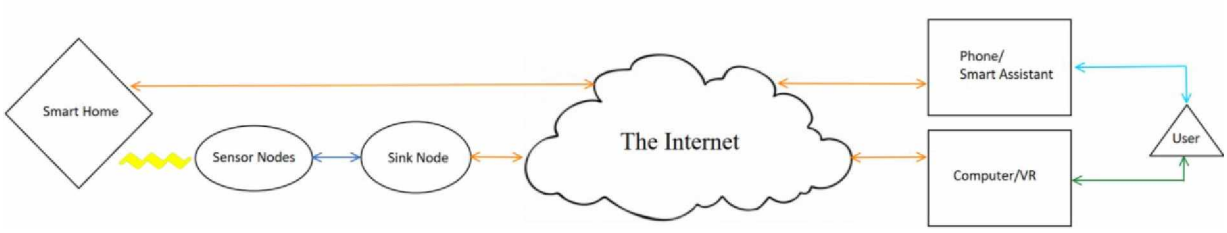


*Figure 80:Online Home Monitoring VR System*

In addition to connecting the system to the Internet, adding the ability for multiple users to interact in the same virtual environment with the same WSN data, as shown in Figure 81, would be a valuable feature for applications where collaboration was necessary. This modification would not be very difficult to implement, as the Unreal Engine 4 IDE has built in multi-user functionality that can be applied to the VR program.

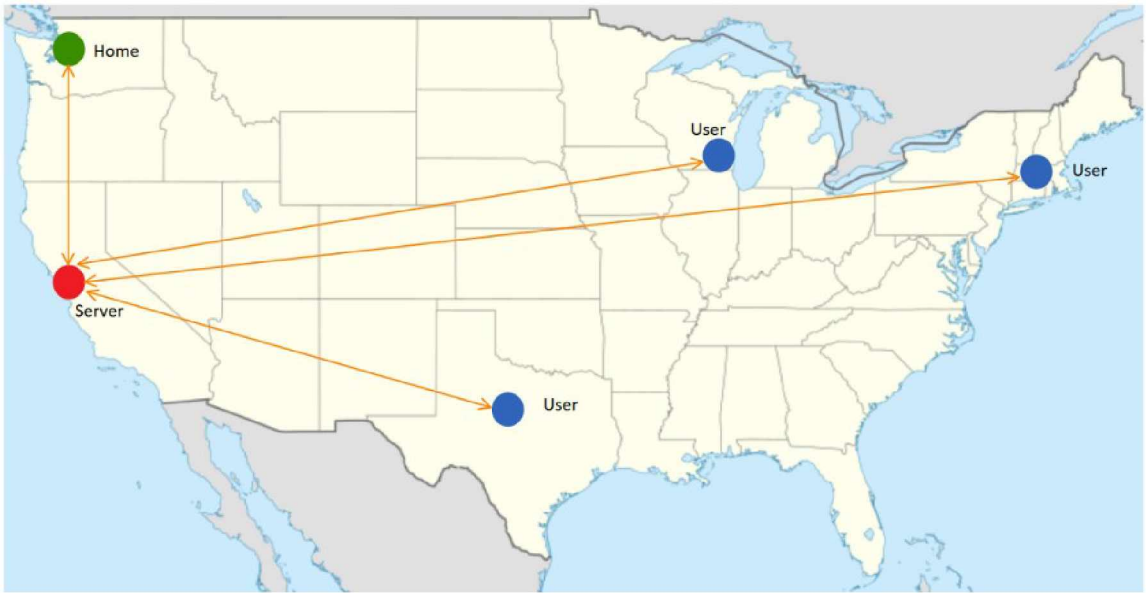


*Figure 81: Online Multi User Home Monitoring VR System*

The potential applications of the system developed in this paper are not limited to VR but can be applied to AR systems as well. This thesis demonstrated that the Home Monitoring VR system stands as a valuable and feature rich system, as a platform for future research, and as inspiration for future researchers and engineers to integrate VR, smart home, and wireless senor network technology into state-of-the-art applications.

## References


[1] "History of Virtual Reality," Virtual Reality Society, [Online]. Available: https://www.vrs.org.uk/virtual-reality/history.html. [Accessed 26 Mar. 2019].

[2] H. Costello, "Global Virtual Reality Market Forcase 2020 by Major Players such as Sony, Microsoft, Facebook, HTC, Google, Samsung Electronics, GoPro, etc.," Reuters, 24 Apr. 2017. [Online]. Available: https://www.reuters.com/brandfeatures/venture-capital/article?id=4975. [Accessed 26 Mar. 2019].

[3] "VR Technology Helps Paraplegics Regain Leg Function," Business Insider Intelligence, 15 Aug. 2016. [Online]. Available: https://www.businessinsider.com/vr-technology-helps-paraplegics-regain-leg-function-2016-8. [Accessed 26 Mar. 2019].

[4] B. Krishnamachari, Networking Wireless Sensors, New York: Cambridge University Press, 2005.

[5] J. Arora, G. Deep and R. Kumar, "IoT-Based Smart Home Systems," in *Innovations in Computer Science and Engineering*, Singapore, Springer, 2017, pp. 531-538.

[6] "NEXTVR," NEXTVR, 2018. [Online]. Available: https://nextvr.com/. [Accessed 28 Mar. 2019].

[7] L. Matney, "How to live stream today's Oculus Connect 5 keynote online and in VR," Tech Crunch, Sep. 2018. [Online]. Available: https://techcrunch.com/2018/09/26/how-to-livestream-todays-oculus-connect-5-keynote-online-and-in-vr/. [Accessed 28 Mar. 2019].

[8] B. Lang, "Yes, You Can Juggle in VR with the HTC Vive," Road To VR, 14 May 2015. [Online]. Available: https://www.roadtovr.com/yes-you-can-juggle-in-vr-with-the-htc-vive/. [Accessed 28 Mar. 2019].

[9] "Keep Talking and Noboyd Explodes," Steel Crate Games Inc., [Online]. Available: http://www.keeptalkinggame.com. [Accessed 28 Mar. 2019].

[10] "Star Trek: Bridge Crew," UBISOFT, 2017. [Online]. Available: https://www.ubisoft.com/en-us/game/star-trek-bridge-crew/. [Accessed 28 Mar. 2019].

[11] V. Pitre, "Top 7 Business Collaboration Conference Apps in Virtual Reality (VR)," VU Dream, 7 Dec. 2017. [Online]. Available: http://www.vudream.com/top-7-business-conference-apps-virtual-reality-vr/. [Accessed 28 Mar. 2019].

[12] "VFX1 Headgear Requirements," Mindflux, 2006. [Online]. Available: http://www.mindflux.com.au/products/iis/vfx1-2.html#requirements. [Accessed 28 Mar. 2019].

[13] "Home Computers and Internet Use in the United States: August 2000," Bureau, U.S. Census, 2001.

[14] C. Ryan and J. M. Lewis, "Computer and Internet Use in the United States 2015," U.S. Census Bureau, 2017.

[15] K. Chaykowski, "Inside Facebook's Bet On An Augmented Reality Future," Forbes, 8 Mar. 2018. [Online]. Available: https://www.forbes.com/sites/kathleenchaykowski/2018/03/08/inside-facebooks-bet-on-an-augmented-reality-future/#737ec2d94d56. [Accessed 28 Mar. 2019].

[16] "Virtual Reality (VR): Global Market Forecasts to 2023 - Exponential Growth of 49.71% CAGR is Expected," Business Wire, 3 May 2018. [Online]. Available: https://www.businesswire.com/news/home/20180503005628/en/Virtual-Reality-VR-Global-Market-Forecasts-2023. [Accessed 28 Mar. 2019].

[17] J. Kite-Powell, "See How This Company Uses Virtual Reality To Change Patient Healthcare," Forbes, 30 Sep. 2018. [Online]. Available: https://www.forbes.com/sites/jenniferhicks/2018/09/30/see-how-this-company-uses-virtual-reality-to-change-patient-healthcare/#64dc3e93455e. [Accessed 28 Mar. 2019].

[18] "The global market for virtual reality technologies totaled $3.7 billion in 2017 and should reach $39.4 billion by 2022, growing at a compound annual growth rate (CAGR) or 60.5% during 2017-2022," PR Newswire, 05 Feb. 2018. [Online]. Available: https://www.prnewswire.com/news-releases/the-global-market-for-virtual-reality-technologies-totaled-37-billion-in-2017-and-should-reach-394-billion-by-2022-growing-at-a-compound-annual-growth-rate-cagr-of-605-during-2017-2022-300593341.html. [Accessed 28 Mar. 2019].

[19] S. Kroc and V. Delic, "Personal wireless sensor network for mobile health care monitoring," in *6th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Service*, 2003.

[20] "Wireless Sensor Networks and their Applications," ELPROCUS, [Online]. Available: https://www.elprocus.com/introduction-to-wireless-sensor-networks-types-and-applications/. [Accessed 28 Mar. 2019].

[21] "Google Scholar Results," Google, 2018. [Online]. Available: https://scholar.google.com/scholar?as_ylo=2018&q=wireless+sensor+networks&hl=en&as_sdt=0,48. [Accessed 28 Mar. 2019].

[22] P. P. Gaikwad, J. P. Gabhane and S. S. Golait, "A Survey Based on Smart Homes System Using Internet-of-Things," in *2015 International Conference on Computation of Power, Energy, Information and Communication*, 2015.

[23] M. Kocakulak and I. Butun, "An Overview of Wireless Sensor Networks Towards Internet of Things," in *IEEE 7th Annual Computing and Communication Workshop and Conference*, Las Vegas, 2017.

[24] "10 Popular Mobile Digital Assistants," Software Development Services, 2019. [Online]. Available: https://www.flatworldsolutions.com/IT-services/articles/popular-digital-assistants.php. [Accessed 28 Mar. 2019].

[25] L. Reese, "The Home Invasion of Smart Technology - an emerging Market," Mouser Electronics, [Online]. Available: https://www.mouser.com/applications/smart_home_technologies/. [Accessed 28 Mar. 2019].

[26] J. Feltham, "Mixing Realities, True Haptics And Photorealistic Humans: 5 Big Takeaways From Michael Abrash's OC5 Keynote," Upload VR, 2 Oct. 2018. [Online]. Available: https://uploadvr.com/abrash-2018-predictions-oc5/. [Accessed 28 Mar 2019].

[27] R. C. Dorf, "The Networked Embedded System Revolution," in *The Electrical Engineering Handbook, Third Edition: Systems, Controls, Embedded Systems, Energy, and Machines*, Boca Raton, CRC Press, 2006, pp. 16-15 - 16-24.

[28] I. M. M. El Emary and S. Ramakrishnan, Wireless Sensor Networks From Theory to Applications, Boca Raton: CRC Press, 2014.

[29] S.-H. Yang, Wireless Sensor Networks Principles, Design and Applications, London: Springer-Verlag, 2014.

[30] B. Kan, L. Cail, L. Zhao and Y. Xu, "Energy Efficient Design of WSN Based on an Accurate Power Consumption Model," in *International Conference on Wireless Communications, Networking and Mobile Computing*, 2007.

[31] K. Nair, J. Kulkarni, M. Warde, Z. Dave, V. Rawalgaonkar, G. Gore and J. Joshi, "Optimizing Power Consumption on IoT based Wireless Sensor Networks Using Bluetooth Low Energy," in *International Conference on Green Computing and Internet of Things*, Noida, 2015.

[32] Q. Wang, M. Hempstead and W. Yang, "A Realistic Power Consumption Model for Wireless Sensor Network Devices," in *3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, 2006.

[33] B. Dushime, "Design and Implementation of Application-Specific Medium Access Control Protocol for Scalable Smart Home Embedded Systems," Fairbanks, 2016.

[34] L. Jiang, D.-Y. Liu and B. Yang, "Smart Home Research," in *International Conference on Machine Learning and Cybernetics*, 2004.

[35] S. Mare, L. Girvin, F. Roesner and T. Kohno, "Consumer Smart Homes: Where We Are and Where We Need to Go," 2019.

[36] C. Gomez, S. Chessa, A. Fleury, G. Roussos and D. Preuceneers, "Internet of Things for Enabling Smart Environments: A Technology-Centric Perspective," *Journal of Ambient Intelligence and Smart Environments,* pp. 23-43, 2019.

[37] "Shop By Category | Smart Home," Amazon, [Online]. Available: https://www.amazon.com/smart-home-devices/b?ie=UTF8&node=6563140011. [Accessed 28 Mar. 2019].

[38] V. Chattaraman, W.-S. Kwon, J. E. Gilbert and K. Ross, "Should AI-Based, Conversational Digital Assistants Employ Social- or Task-oriented interaction style? A Task-Competency and Reciprocity Perspective for Older Adults," *Computers in Human Behavior,* vol. 90, pp. 315-330, 2019.

[39] G. C. Burdea and P. Coiffet, Virtual Reality Technology Second Edition, Hoboken: Wiley & Sons, 2003.

[40] L. P. Berg and J. M. Vance, "Industry Use of Virtual Reality in Product Design and Manufacturing: A Survey," *Virtual Reality,* vol. 21, no. 1, pp. 1-17, 2017.

[41] W. R. Sherman and A. B. Craig, Understanding Virtual Reality: Interface, Application, and Design Second Edition, Cambridge: Elsevier, 2019.

[42] W. Greenwald, "The Best VR (Virtual Reality) Headsets for 2019," PCMag, 1 Mar. 2019. [Online]. Available: https://www.pcmag.com/article/342537/the-best-virtual-reality-vr-headsets. [Accessed 22 Apr. 2019].

[43] S. Hayden, "Oculus Rift S Vs. Oculus Quest - Specs & Features," Road To VR, 20 Mar. 2019. [Online]. Available: https://www.roadtovr.com/oculus-rift-s-vs-quest-specs-difference/. [Accessed 22 Apr. 2019].

[44] "Daydream," Google, [Online]. Available: https://vr.google.com/daydream/. [Accessed 22 Apr. 2019].

[45] "Google Cardboard," Google, [Online]. Available: https://vr.google.com/cardboard/. [Accessed 22 Apr. 2019].

[46] D. Goldsmith, F. Liarokapis, G. Malone and J. Kemp, "Augmented Reality Environmental Monitoring Using Wireless Sensor Networks," in *International Conference Information Visualization*, London, 2008.

[47] K. Sato, S. Naoya and S. Hideki, "Visualization and Management Platform with Augmented Reality for Wireless Sensor Networks.," *Wireless Sensor Network,* vol. 7, pp. 1-11, 2015.

[48] Brad, "VR for IoT - Rethinking Data Visualization," Eye Create Worlds, 12 Jan. 2016. [Online]. Available: http://www.eyecreateworlds.com/index.php/56-vr-for-iot-and-pushing-the-limits-of-the-dk2-tracking-volume. [Accessed 28 Mar 2019].

[49] "MSP430F5529 LaunchPad Development Kit (MSP-EXP40F5529LP)," Texas Instruments, 2017.

[50] "MSP430F5529," Texas Instruments, 2019. [Online]. Available: http://www.ti.com/product/MSP430F5529?keyMatch=5529&tisearch=Search-EN-Everything#features. [Accessed 9 Apr. 2019].

[51] "TPS6223xx 2-MHz and 3-MHz Ultra Small Step-Down Converter in 1 x 1.5 USON Package," 2016. [Online]. Available: http://www.ti.com/lit/ds/symlink/tps62230.pdf. [Accessed 23 Apr. 2019].

[52] "TUSB2046x 4-Port Hub for the Universal Serial Bus With Optional Serial EEPROM Interface," Jun. 2017. [Online]. Available: http://www.ti.com/lit/ds/symlink/tusb2046b.pdf. [Accessed 23 Apr. 2019].

[53] "+/- 2g Tri-axis Analog Accelerometer Specifications," Kionix, Ithaca, 2015.

[54] "Infrared Thermopile Sensor in Chip-Scale Package," Texas Instruments, 2012.

[55] P. Schlyter, "Starlight Illuminance Coincides with the Human Eye's Minimum Illuminance While Moonlight Coincides with the Human Eye's Minimum Colour Vision Illuminance," *IEE Reviews,* p. 1183, 1972.

[56] "Illuminance - Recommended Light Level," Engineering ToolBox, 2004. [Online]. Available: https://www.engineeringtoolbox.com/light-level-rooms-d_708.html. [Accessed 10 Apr. 2019].

[57] "OPT3001 Ambient Light Sensor (ALS)," Texas Instruments, 2017.

[58] "Cree PLCC4 3 in 1 SMD LED CLV1A-FKB," Cree, Durham, 2018.

[59] "Part No: CEM-1203(42) Description: Magnetic Buzzer," CUI INC, Tualatin, 2006.

[60] "nRF24L01+ Single Chip 2.4 GHz Transceiver Product Specification v1.0," Nordic Semiconductor, Trondheim, 2008.

[61] "2.4G Dipole 2dBi Antenna," Chang Hong Technology Co., Ltd..

[62] "C by GE," General Electric Company, 2019. [Online]. Available: https://www.cbyge.com/pages/pdp-life. [Accessed 10 Apr. 2019].

[63] "Oculus," Oculus, [Online]. Available: www.oculus.com. [Accessed 26 Mar. 2019].

[64] "Oculus Rift CV1 Teardown," iFixit, 30 Mar. 2016. [Online]. Available: https://www.ifixit.com/Teardown/Oculus+Rift+CV1+Teardown/60612. [Accessed 10 Apr. 2019].

[65] "Oculus Touch Teardown," iFixIt, 7 Dec. 2016. [Online]. Available: https://www.ifixit.com/Teardown/Oculus+Touch+Teardown/75109. [Accessed 10 Apr. 2019].

[66] "Google Maps, 2019. UAF, [online] Accessed Apr. 7 2019," 2019.

[67] "MSP430F552x, MSP430F552x Mixed-Signal Microcontrollers," Texas Instruments, 2018.

[68] "BOOSTXL-EDUMKII Educational BoosterPack Plug-in Module Mark II," Texas Instruments, 2017.

# Appendices

## Appendix A: Texas Instruments MSP-EXP430F5529LP Block Diagram

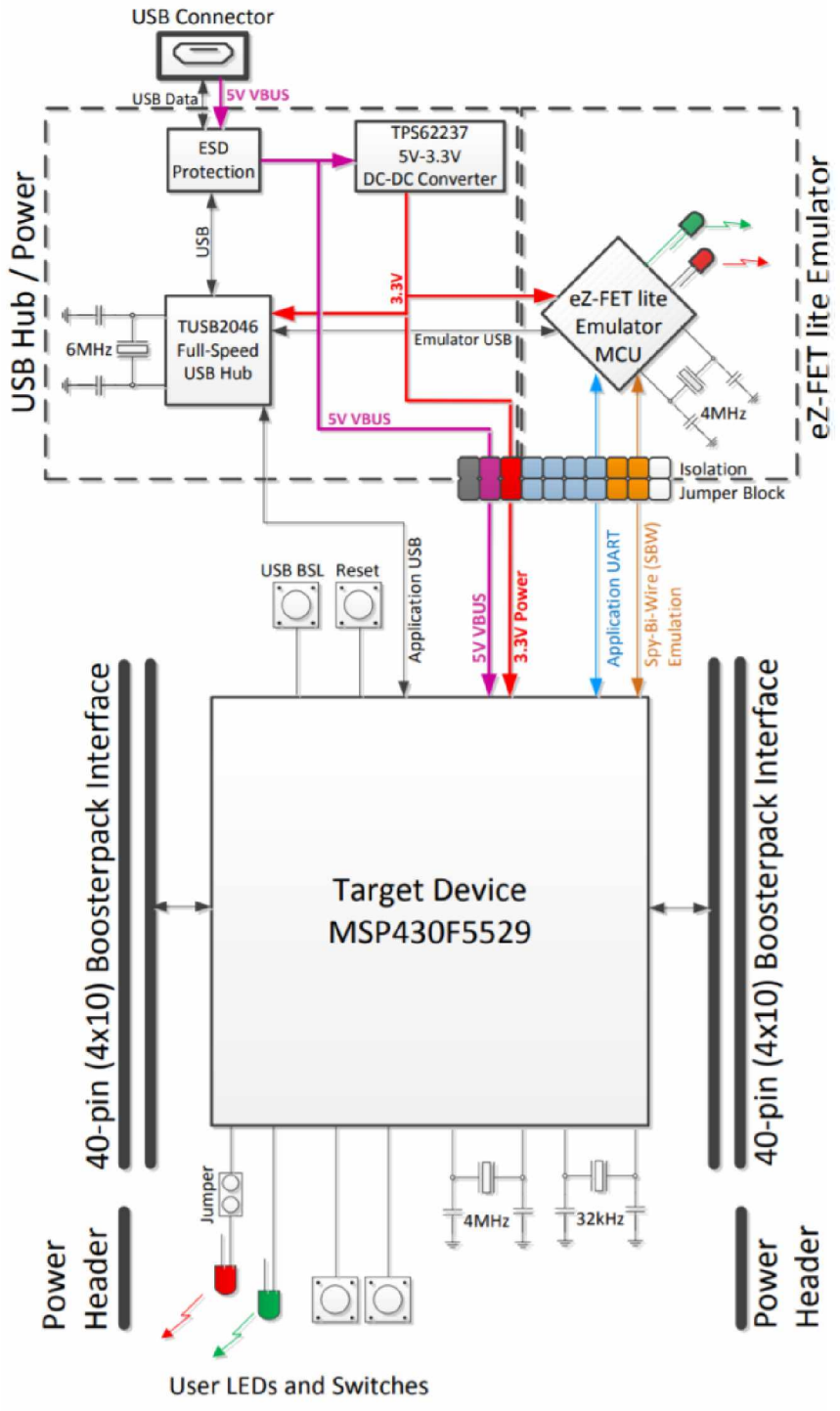


*Figure 82: TI MSP-EXP430F5529LP Block Diagram [49]*

## Appendix B: MSP430F5529 Microcontroller Documentation

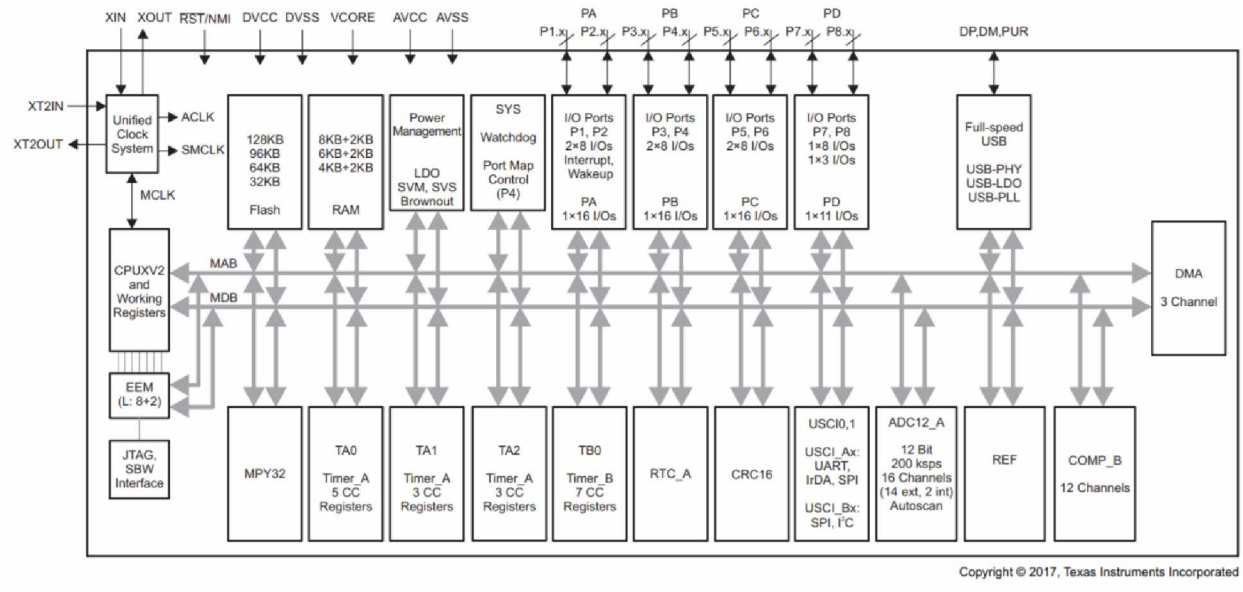


*Figure 83: MSP430F5529 Functional Block Diagram [67]*

*Table 19: Active Mode Supply Current into Vcc Excluding External Current [67]*

over recommended operating free-air temperature (unless otherwise noted)(1) (2) (3)

| PARAMETER | EXECUTION MEMORY | $V_{CC}$ | PMMCOREVx | FREQUENCY ($f_{DCO} = f_{MCLK} = f_{SMCLK}$) | | | | | | | | | | UNIT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 1 MHz | | 8 MHz | | 12 MHz | | 20 MHz | | 25 MHz | | |
| | | | | TYP | MAX | TYP | MAX | TYP | MAX | TYP | MAX | TYP | MAX | |
| $I_{AM, Flash}$ | Flash | 3.0 V | 0 | 0.36 | 0.47 | 2.32 | 2.60 | | | | | | | mA |
| | | | 1 | 0.40 | | 2.65 | | 4.0 | 4.4 | | | | | |
| | | | 2 | 0.44 | | 2.90 | | 4.3 | | 7.1 | 7.7 | | | |
| | | | 3 | 0.46 | | 3.10 | | 4.6 | | 7.6 | | 10.1 | 11.0 | |
| $I_{AM, RAM}$ | RAM | 3.0 V | 0 | 0.20 | 0.24 | 1.20 | 1.30 | | | | | | | mA |
| | | | 1 | 0.22 | | 1.35 | | 2.0 | 2.2 | | | | | |
| | | | 2 | 0.24 | | 1.50 | | 2.2 | | 3.7 | 4.2 | | | |
| | | | 3 | 0.26 | | 1.60 | | 2.4 | | 3.9 | | 5.3 | 6.2 | |

(1) All inputs are tied to 0 V or to $V_{CC}$. Outputs do not source or sink any current.
(2) The currents are characterized with a Micro Crystal MS1V-T1K crystal with a load capacitance of 12.5 pF. The internal and external load capacitance are chosen to closely match the required 12.5 pF.
(3) Characterized with program executing typical data processing. USB disabled (VUSBEN = 0, SLDOEN = 0).
$f_{ACLK}$ = 32786 Hz, $f_{DCO} = f_{MCLK} = f_{SMCLK}$ at specified frequency.
XTS = CPUOFF = SCG0 = SCG1 = OSCOFF= SMCLKOFF = 0.

*Table 20: Low-Power Mode Supply Currents (into Vcc) Excluding External Current [67]*

over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted)[1] [2]

| PARAMETER | | $V_{CC}$ | PMMCOREVx | –40°C | | 25°C | | 60°C | | 85°C | | UNIT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | TYP | MAX | TYP | MAX | TYP | MAX | TYP | MAX | |
| $I_{LPM0,1MHz}$ | Low-power mode 0[3][4] | 2.2 V | 0 | 73 | | 77 | 85 | 80 | | 85 | 97 | µA |
| | | 3.0 V | 3 | 79 | | 83 | 92 | 88 | | 95 | 105 | |
| $I_{LPM2}$ | Low-power mode 2[5][4] | 2.2 V | 0 | 6.5 | | 6.5 | 12 | 10 | | 11 | 17 | µA |
| | | 3.0 V | 3 | 7.0 | | 7.0 | 13 | 11 | | 12 | 18 | |
| $I_{LPM3,XT1LF}$ | Low-power mode 3, crystal mode[6][4] | 2.2 V | 0 | 1.60 | | 1.90 | | 2.6 | | 5.6 | | µA |
| | | | 1 | 1.65 | | 2.00 | | 2.7 | | 5.9 | | |
| | | | 2 | 1.75 | | 2.15 | | 2.9 | | 6.1 | | |
| | | 3.0 V | 0 | 1.8 | | 2.1 | 2.9 | 2.8 | | 5.8 | 8.3 | |
| | | | 1 | 1.9 | | 2.3 | | 2.9 | | 6.1 | | |
| | | | 2 | 2.0 | | 2.4 | | 3.0 | | 6.3 | | |
| | | | 3 | 2.0 | | 2.5 | 3.9 | 3.1 | | 6.4 | 9.3 | |
| $I_{LPM3,VLO}$ | Low-power mode 3, VLO mode[7][4] | 3.0 V | 0 | 1.1 | | 1.4 | 2.7 | 1.9 | | 4.9 | 7.4 | µA |
| | | | 1 | 1.1 | | 1.4 | | 2.0 | | 5.2 | | |
| | | | 2 | 1.2 | | 1.5 | | 2.1 | | 5.3 | | |
| | | | 3 | 1.3 | | 1.6 | 3.0 | 2.2 | | 5.4 | 8.5 | |
| $I_{LPM4}$ | Low-power mode 4[8][4] | 3.0 V | 0 | 0.9 | | 1.1 | 1.5 | 1.8 | | 4.8 | 7.3 | µA |
| | | | 1 | 1.1 | | 1.2 | | 2.0 | | 5.1 | | |
| | | | 2 | 1.2 | | 1.2 | | 2.1 | | 5.2 | | |
| | | | 3 | 1.3 | | 1.3 | 1.6 | 2.2 | | 5.3 | 8.1 | |
| $I_{LPM4.5}$ | Low-power mode 4.5[9] | 3.0 V | | 0.15 | | 0.18 | 0.35 | 0.26 | | 0.5 | 1.0 | µA |

(1) All inputs are tied to 0 V or to $V_{CC}$. Outputs do not source or sink any current.
(2) The currents are characterized with a Micro Crystal MS1V-T1K crystal with a load capacitance of 12.5 pF. The internal and external load capacitance are chosen to closely match the required 12.5 pF.
(3) Current for watchdog timer clocked by SMCLK included. ACLK = low frequency crystal operation (XTS = 0, XT1DRIVEx = 0).
CPUOFF = 1, SCG0 = 0, SCG1 = 0, OSCOFF = 0 (LPM0); $f_{ACLK}$ = 32768 Hz, $f_{MCLK}$ = 0 MHz, $f_{SMCLK}$ = $f_{DCO}$ = 1 MHz
USB disabled (VUSBEN = 0, SLDOEN = 0).
(4) Current for brownout, high-side supervisor ($SVS_H$) normal mode included. Low-side supervisor and monitor disabled ($SVS_L$, $SVM_L$).
High-side monitor disabled ($SVM_H$). RAM retention enabled.
(5) Current for watchdog timer and RTC clocked by ACLK included. ACLK = low frequency crystal operation (XTS = 0, XT1DRIVEx = 0).
CPUOFF = 1, SCG0 = 0, SCG1 = 1, OSCOFF = 0 (LPM2); $f_{ACLK}$ = 32768 Hz, $f_{MCLK}$ = 0 MHz, $f_{SMCLK}$ = $f_{DCO}$ = 0 MHz; DCO setting = 1 MHz operation, DCO bias generator enabled.
USB disabled (VUSBEN = 0, SLDOEN = 0)
(6) Current for watchdog timer and RTC clocked by ACLK included. ACLK = low frequency crystal operation (XTS = 0, XT1DRIVEx = 0).
CPUOFF = 1, SCG0 = 1, SCG1 = 1, OSCOFF = 0 (LPM3); $f_{ACLK}$ = 32768 Hz, $f_{MCLK}$ = $f_{SMCLK}$ = $f_{DCO}$ = 0 MHz
USB disabled (VUSBEN = 0, SLDOEN = 0)
(7) Current for watchdog timer and RTC clocked by ACLK included. ACLK = VLO.
CPUOFF = 1, SCG0 = 1, SCG1 = 1, OSCOFF = 0 (LPM3); $f_{ACLK}$ = $f_{VLO}$, $f_{MCLK}$ = $f_{SMCLK}$ = $f_{DCO}$ = 0 MHz
USB disabled (VUSBEN = 0, SLDOEN = 0)
(8) CPUOFF = 1, SCG0 = 1, SCG1 = 1, OSCOFF = 1 (LPM4); $f_{DCO}$ = $f_{ACLK}$ = $f_{MCLK}$ = $f_{SMCLK}$ = 0 MHz
USB disabled (VUSBEN = 0, SLDOEN = 0)
(9) Internal regulator disabled. No data retention.
CPUOFF = 1, SCG0 = 1, SCG1 = 1, OSCOFF = 1, PMMREGOFF = 1 (LPM4.5); $f_{DCO}$ = $f_{ACLK}$ = $f_{MCLK}$ = $f_{SMCLK}$ = 0 MHz

## Appendix C: Texas Instruments BOOSTXL-EDUMKII Hardware Overview

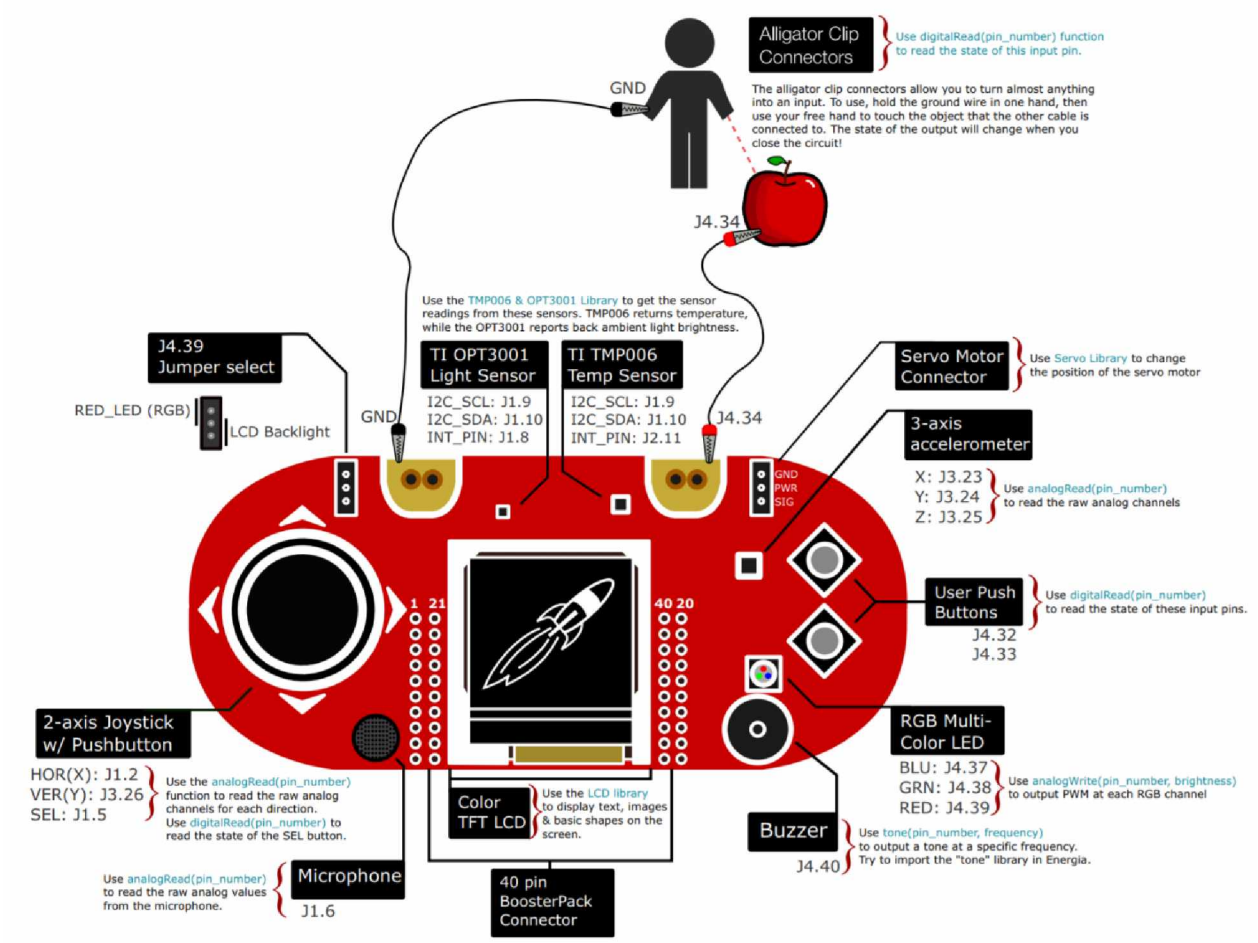


*Figure 84: TI BOOSTXL-EDUMKII Hardware Overview [68]*

## Appendix D: Kionix KXC9-2050 3-Axis Analog Accelerometer Documentation

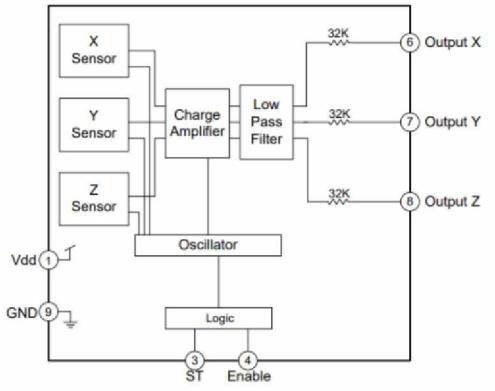


*Figure 85: 3-Axis Accelerometer Functional Diagram [53]*

*Table 21: 3-Axis Accelerometer Mechanical Specifications [53]*

(specifications are for operation at 3.3V and T = 25C unless stated otherwise)

| Parameters | Units | Min | Typical | Max |
|---|---|---|---|---|
| Operating Temperature Range | °C | -40 | - | 85 |
| Zero-g Offset | V | 1.567 | 1.65 | 1.732 |
| Zero-g Offset Variation from RT over Temp. | mg/°C | | 0.7 (xy)<br>0.4 (z) | |
| Sensitivity | mV/g | 640 | 660 | 680 |
| Sensitivity Variation from RT over Temp. | %/°C | | 0.01 (xy)<br>0.04 (z) | |
| Offset Ratiometric Error ($V_{dd}$ = 3.3V ± 5%) | % | | 0.2 | |
| Sensitivity Ratiometric Error ($V_{dd}$ = 3.3V ± 5%) | % | | 0.3 (xy)<br>0.15 (z) | |
| Self Test Output change on Activation | g | | 2.9 (x)<br>2.8 (y)<br>1.9 (z) | |
| Mechanical Resonance (-3dB)[1] | Hz | | 3500 (xy)<br>1800 (z) | |
| Non-Linearity | % of FS | | 0.1 | |
| Cross Axis Sensitivity | % | | 2 | |
| Noise Density (on filter pins) | µg / √Hz | | 125 | |

Notes:

1. Resonance as defined by the dampened mechanical sensor.

*Table 22: 3-Axis Accelerometer Electrical Specifications [53]*

(specifications are for operation at 3.3V and T = 25C unless stated otherwise)

| Parameters | | Units | Min | Typical | Max |
|---|---|---|---|---|---|
| Supply Voltage ($V_{dd}$) | Operating | V | 1.8 | 3.3 | 3.6 |
| Current Consumption | Operating (full power) | µA | 170 | 240 | 310 |
| | Standby | µA | | 5 | |
| Analog Output Resistance($R_{out}$) | | kΩ | 24 | 32 | 40 |
| Power Up Time[1] | | ms | - | 5*$R_{out}$*C | - |
| Bandwidth (-3dB)[2] | | Hz | 40 | 50 | 60 |

Notes:

1. Power up time is determined by 5 times the RC time constant of the factory programmed or user defined low pass filter.

2. Factory programmable to have a switched capacitor low pass filter at 2kHz, 1kHz, 500Hz, 100Hz, 50Hz, or no low pass filter. Optionally, the user can define with external capacitors. Maximum defined by the frequency response of the sensors.

# Appendix E: Texas Instruments TMP006 Thermopile Sensor Documentation

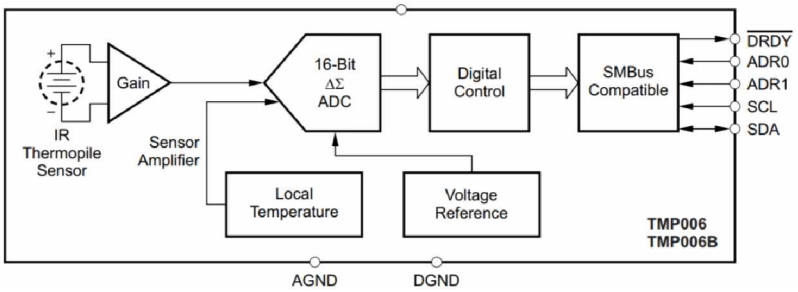


*Figure 86: TI TMP006 Thermopile Sensor Functional Block Diagram [54]*

*Table 23: TI TMP006 Thermopile Sensor Electrical Characteristics [54]*

At $T_A$ = +25°C, V+ = 3.3 V, and conversion time = 1 second, unless otherwise specified.

| PARAMETER | | TEST CONDITIONS | MIN | TYP | MAX | UNIT |
|---|---|---|---|---|---|---|
| **OUTPUT ERROR** | | | | | | |
| Ambient temperature sensor | | $T_A$ = 0°C to +60°C, V+ = 2.2 V to 5.5 V | | ±0.5 | ±1 | °C |
| | | $T_A$ = –40°C to +125°C, V+ = 2.2 V to 5.5 V | | ±0.5 | ±1.5 | °C |
| Power-supply rejection ratio | PSRR | | | 0.1 | | °C/V |
| Sensor voltage | | $T_{Object}$ = +40°C to +60°C, $T_A$ = 0°C to +60°C | | 7 | | µV/°C |
| Calculate object temperature[1] | | $T_A$ = +20°C to +60°C, $T_{Object} - T_A$ = –10°C to +30°C | | ±1 | ±3 | °C |
| Field of view | | 50% responsivity | | 90 | | Degrees |
| **TEMPERATURE MEASUREMENT** | | | | | | |
| Conversion time | | CR2 = 0, CR1 = 0, CR0 = 0 | | 0.25 | | Seconds |
| | | CR2 = 0, CR1 = 0, CR0 = 1 | | 0.5 | | Seconds |
| | | CR2 = 0, CR1 = 1, CR0 = 0 | | 1 | | Seconds |
| | | CR2 = 0, CR1 = 1, CR0 = 1 | | 2 | | Seconds |
| | | CR2 = 1, CR1 = 0, CR0 = 0 | | 4 | | Seconds |
| Resolution | | | | | | |
| Local temperature sensor | | | | 0.03125 | | °C |
| Thermopile sensor resolution | | | | 156.25 | | nV |
| **SMBus COMPATIBLE INTERFACE** | | | | | | |
| Logic input high voltage (SCL, SDA) | $V_{IH}$ | TMP006 only | 2.1 | | | V |
| | | TMP006B only | 1.4 | | | V |
| Logic input low voltage (SCL, SDA) | $V_{IL}$ | TMP006 only | | | 0.8 | V |
| | | TMP006B only | | | 0.4 | V |
| Hysteresis | | | | 100 | | mV |
| Output low voltage (SDA) | $V_{OL}$ | $I_{OUT}$ = 6 mA | | 0.15 | 0.4 | V |
| Output low sink current (SDA) | | | 6 | | | mA |
| Logic input current | | Forced to 0.4 V | –1 | | +1 | µA |
| Input capacitance (SCL, SDA, A0, A1) | | | | 3 | | pF |
| Clock frequency | | | 0.001 | | 3.4 | MHz |
| Interface timeout | | | 25 | 30 | 35 | ms |
| **DIGITAL OUTPUTS** | | | | | | |
| Output low voltage (DRDY pin) | $V_{OL}$ | $I_{OUT}$ = 4 mA | | 0.15 | 0.4 | V |
| High-level output leakage current | $I_{OH}$ | $V_{OUT}$ = $V_{DD}$ | | 0.1 | 1 | µA |
| Output low sink current (DRDY) | | Forced to 0.4 V | 4 | | | mA |
| **POWER SUPPLY** | | | | | | |
| Power-on reset | V+ | T = –40°C to +125°C | | 1.6 | | V |
| Specified voltage range | V+ | T = –40°C to +125°C | 2.2 | | 5.5 | V |
| Quiescent current | $I_Q$ | Continuous conversion; see Table 9 | | 240 | 325 | µA |
| | | Serial bus inactive, shutdown mode, TMP006 only | | 0.5 | 1.0 | µA |
| | | Serial bus inactive, shutdown mode, TMP006B only | | 1.5 | 5.0 | µA |
| | | Serial bus active, $f_S$ = 400 kHz, shutdown mode | | 90 | | µA |
| **TEMPERATURE RANGE** | | | | | | |
| Specified range | | | –40 | | +125 | °C |
| Storage range | | | –65 | | +150 | °C |

(1) This parameter is tested in a fully-settled setup with no transients, in front of an ideal black body, with specified layout constraints, and after system calibration.

# Appendix F: Texas Instruments OPT3001 Ambient Light Sensor Documentation

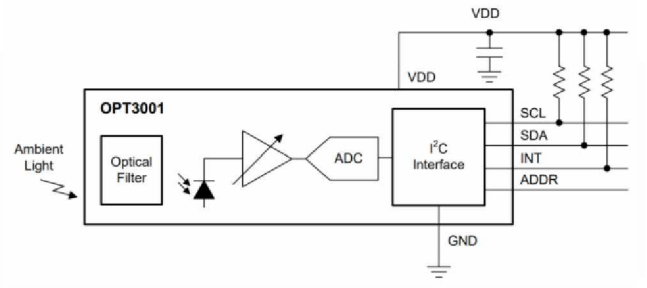


*Figure 87: TI OPT3001 Ambient Light Sensor Functional Block Diagram [57]*

*Table 24: TI OPT3001 Ambient Light Sensor Electrical Characteristics [57]*

At $T_A$ = 25°C, $V_{DD}$ = 3.3 V, 800-ms conversion time (CT = 1)(1), automatic full-scale range (RN[3:0] = 1100b(1)), white LED, and normal-angle incidence of light, unless otherwise specified.

| PARAMETER | | TEST CONDITIONS | | MIN | TYP | MAX | UNIT |
|---|---|---|---|---|---|---|---|
| **OPTICAL** | | | | | | | |
| | Peak irradiance spectral responsivity | | | | 550 | | nm |
| | Resolution (LSB) | Lowest full-scale range, RN[3:0] = 0000b(1) | | | 0.01 | | lux |
| | Full-scale illuminance | | | | 83865.6 | | lux |
| | Measurement output result | 0.64 lux per ADC code, 2620.80 lux full-scale (RN[3:0] = 0110)(1), 2000 lux input(2) | | 2812 | 3125 | 3437 | ADC codes |
| | | | | 1800 | 2000 | 2200 | lux |
| | Relative accuracy between gain ranges(3) | | | | 0.2% | | |
| | Infrared response (850 nm)(2) | | | | 0.2% | | |
| | Light source variation (incandescent, halogen, fluorescent) | Bare device, no cover glass | | | 4% | | |
| | Linearity | Input illuminance > 40 lux | | | 2% | | |
| | | Input illuminance < 40 lux | | | 5% | | |
| | Measurement drift across temperature | Input illuminance = 2000 lux | | | 0.01 | | %/°C |
| | Dark condition, ADC output | 0.01 lux per ADC code | | | 0 | 3 | ADC codes |
| | | | | | 0 | 0.03 | lux |
| | Half-power angle | 50% of full-power reading | | | 47 | | degrees |
| PSRR | Power-supply rejection ratio | $V_{DD}$ at 3.6 V and 1.6 V | | | 0.1 | | %/V(4) |
| **POWER SUPPLY** | | | | | | | |
| $V_{DD}$ | Operating range | | | 1.6 | | 3.6 | V |
| $V_{PC}$ | Operating range of $I^2C$ pull-up resistor | $I^2C$ pull-up resistor, $V_{DD} \le V_{PC}$ | | 1.6 | | 5.5 | V |
| $I_Q$ | Quiescent current | Dark | Active, $V_{DD}$ = 3.6 V | | 1.8 | 2.5 | µA |
| | | | Shutdown (M[1:0] = 00)(1), $V_{DD}$ = 3.6 V | | 0.3 | 0.47 | µA |
| | | Full-scale lux | Active, $V_{DD}$ = 3.6 V | | 3.7 | | µA |
| | | | Shutdown, (M[1:0] = 00)(1) | | 0.4 | | µA |
| POR | Power-on-reset threshold | $T_A$ = 25°C | | | 0.8 | | V |
| **DIGITAL** | | | | | | | |
| | I/O pin capacitance | | | | 3 | | pF |
| | Total integration time(5) | (CT = 1)(1), 800-ms mode, fixed lux range | | 720 | 800 | 880 | ms |
| | | (CT = 0)(1), 100-ms mode, fixed lux range | | 90 | 100 | 110 | ms |
| $V_{IL}$ | Low-level input voltage (SDA, SCL, and ADDR) | | | 0 | | 0.3 × $V_{DD}$ | V |
| $V_{IH}$ | High-level input voltage (SDA, SCL, and ADDR) | | | 0.7 × $V_{DD}$ | | 5.5 | V |
| $I_{IL}$ | Low-level input current (SDA, SCL, and ADDR) | | | | 0.01 | 0.25(6) | µA |
| $V_{OL}$ | Low-level output voltage (SDA and INT) | $I_{OL}$ = 3 mA | | | | 0.32 | V |
| $I_{ZH}$ | Output logic high, high-Z leakage current (SDA, INT) | Pin at $V_{DD}$ | | | 0.01 | 0.25(6) | µA |
| **TEMPERATURE** | | | | | | | |
| | Specified temperature range | | | –40 | | 85 | °C |

(1) Refers to a control field within the configuration register.
(2) Tested with the white LED calibrated to 2k lux and an 850-nm LED.
(3) Characterized by measuring fixed near-full-scale light levels on the higher adjacent full-scale range setting.
(4) PSRR is the percent change of the measured lux output from its current value, divided by the change in power supply voltage, as characterized by results from 3.6-V and 1.6-V power supplies.
(5) The conversion time, from start of conversion until the data are ready to be read, is the integration time plus 3 ms.
(6) The specified leakage current is dominated by the production test equipment limitations. Typical values are much smaller.

## Appendix G: Cree CLV1A-FKB RGB Multicolor LED Documentation

*Table 25: Typical Electrical & Optical Characteristics (Ta = 25°C) [58]*

| Characteristics | Condition | Symbol | Values | | | Unit |
|---|---|---|---|---|---|---|
| | | | R | G | B | |
| Dominant Wavelength | $I_F$ = 20 mA | $\lambda_{DOM}$ | 619~624 | 520~535 | 460~475 | nm |
| Spectral bandwidth at 50% $I_{REL}$ max | $I_F$ = 20 mA | Δ λ | 24 | 38 | 28 | nm |
| Forward Voltage | $I_F$ = 20 mA | $V_{F[avg]}$ | 2.0 | 3.2 | 3.2 | V |
| | | $V_{F(max)}$ | 2.6 | 4.0 | 4.0 | V |
| Luminous Intensity | $I_F$ = 20 mA | $I_{V(min)}$ | 505 | 900 | 224 | mcd |
| | | $I_{V(avg)}$ | 710 | 1450 | 310 | mcd |
| Reverse Current (max) | $V_R$ = 5 V | $I_R$ | 10 | 10 | 10 | μA |

## Appendix H: Custom Routing Daughterboard Schematic and Board Layout

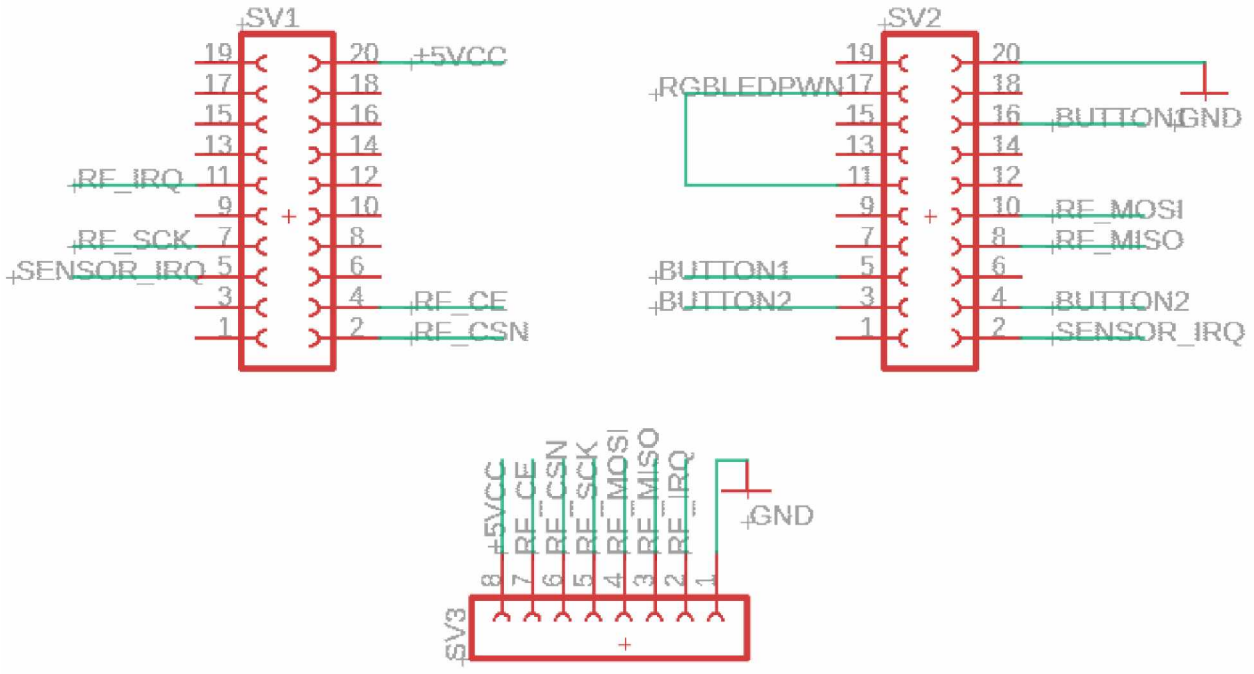


*Figure 88: Custom Routing Daughterboard Schematic*

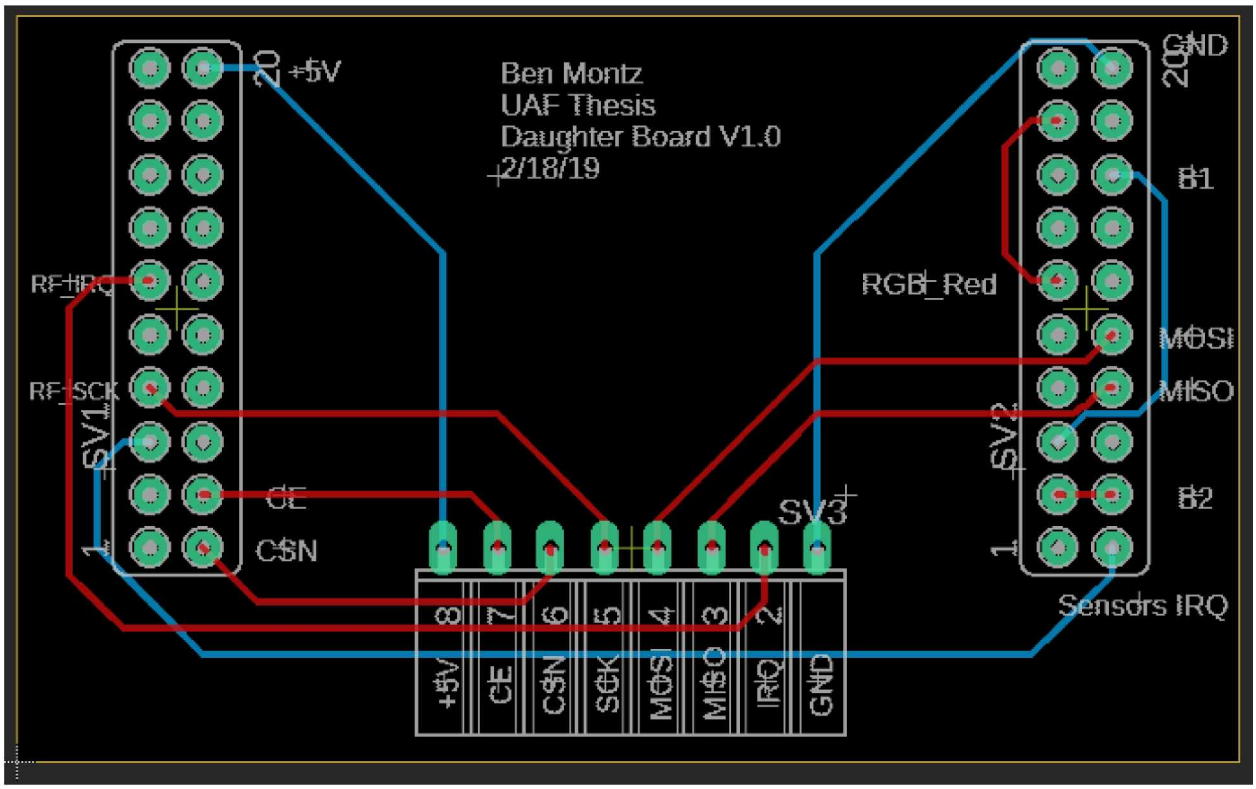


*Figure 89: Custom Routing Daughterboard Layout*

## Appendix I: Nordic Semiconductor nRF24L01+ 2.4 GHz Transceiver Documentation

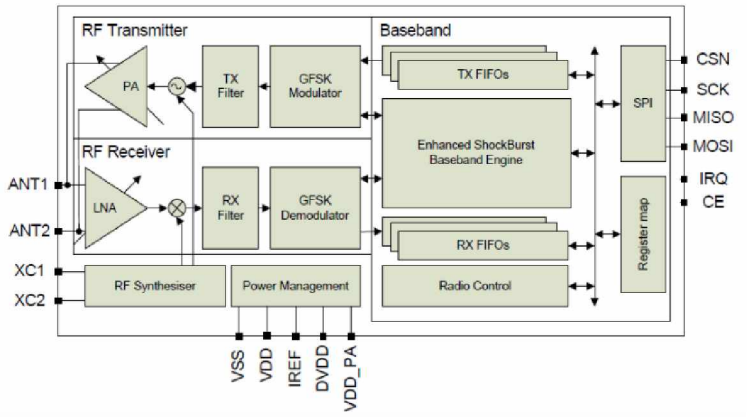


*Figure 90: Nordic Semiconductor nRF24L01+ 2.4 GHz Transceiver Functional Block Diagram [60]*

*Table 26: Nordic Semiconductor nRF24L01+ 2.4 GHz Transceiver Power Consumption Table [60]*

Conditions: VDD = +3V, VSS = 0V, $T_A$ = - 40°C to + 85°C

### Power consumption

| Symbol | Parameter (condition) | Notes | Min. | Typ. | Max. | Units |
|---|---|---|---|---|---|---|
| | **Idle modes** | | | | | |
| $I_{VDD\_PD}$ | Supply current in power down | | | 900 | | nA |
| $I_{VDD\_ST1}$ | Supply current in standby-I mode | a | | 26 | | µA |
| $I_{VDD\_ST2}$ | Supply current in standby-II mode | | | 320 | | µA |
| $I_{VDD\_SU}$ | Average current during 1.5ms crystal oscillator startup | | | 400 | | µA |
| | **Transmit** | | | | | |
| $I_{VDD\_TX0}$ | Supply current @ 0dBm output power | b | | 11.3 | | mA |
| $I_{VDD\_TX6}$ | Supply current @ -6dBm output power | b | | 9.0 | | mA |
| $I_{VDD\_TX12}$ | Supply current @ -12dBm output power | b | | 7.5 | | mA |
| $I_{VDD\_TX18}$ | Supply current @ -18dBm output power | b | | 7.0 | | mA |
| $I_{VDD\_AVG}$ | Average Supply current @ -6dBm output power, ShockBurst™ | c | | 0.12 | | mA |
| $I_{VDD\_TXS}$ | Average current during TX settling | d | | 8.0 | | mA |
| | **Receive** | | | | | |
| $I_{VDD\_2M}$ | Supply current 2Mbps | | | 13.5 | | mA |
| $I_{VDD\_1M}$ | Supply current 1Mbps | | | 13.1 | | mA |
| $I_{VDD\_250}$ | Supply current 250kbps | | | 12.6 | | mA |
| $I_{VDD\_RXS}$ | Average current during RX settling | e | | 8.9 | | mA |

a. This current is for a 12pF crystal. Current when using external clock is dependent on signal swing.
b. Antenna load impedance = 15Ω+j88Ω..
c. Antenna load impedance = 15Ω+j88Ω. Average data rate 10kbps and max. payload length packets.
d. Average current consumption during TX startup (130µs) and when changing mode from RX to TX (130µs).
e. Average current consumption during RX startup (130µs) and when changing mode from TX to RX (130µs).

# Appendix J: Texas Instruments TMP006 Thermopile Sensor Temperature Conversion

## TEMPERATURE FORMAT

The Temperature Register data format of the TMP006 and TMP006B is reported in a binary twos complement signed integer format, as Table 7 shows, with 1 LSB = 1/32°C = 0.03125.

**Table 7. Temperature Data Format**

| TEMPERATURE (°C) | DIGITAL OUTPUT (BINARY) | SHIFTED HEX |
|---|---|---|
| 150 | 0100 1011 0000 0000 | 12CO |
| 125 | 0011 1110 1000 0000 | 0FA0 |
| 100 | 0011 0010 0000 0000 | 0C80 |
| 80 | 0010 1000 0000 0000 | 0A00 |
| 75 | 0010 0101 1000 0000 | 0960 |
| 50 | 0001 1001 0000 0000 | 0640 |
| 25 | 0000 1100 1000 0000 | 0320 |
| 0.03125 | 0000 0000 0000 0100 | 0001 |
| 0 | 0000 0000 0000 0000 | 0000 |
| –0.03125 | 1111 1111 1111 1100 | FFFC |
| –0.0625 | 1111 1111 1111 1000 | FFF8 |
| –25 | 1111 0011 0111 0000 | F370 |
| –40 | 1110 1011 1111 1100 | EBFC |
| –55 | 1110 0100 0111 1100 | E47C |

Converting the integer temperature result of the TMP006 and TMP006B to physical temperature is done by right-shifting the last two LSBs followed by a divide-by-32 of $T_{REG}$ to obtain the physical temperature result in degrees Celsius. $T_{REG}$ is the 14-bit signed integer contained in the corresponding register. The sign of the temperature is the same as the sign of the integer read from the TMP006 and TMP006B. In twos complement notation, the MSB is the sign bit. If the MSB is '1', the integer is negative and the absolute value can be obtained by inverting all bits and adding '1'. An alternative method of calculating the absolute value of negative integers is abs(i) = i xor FFFFh + 1.

*Figure 91: TI TMP006 Temperature Conversion [54]*

# Appendix K: Texas Instruments OPT3001 Ambient Light Sensor Brightness Conversion

**Figure 23. Result Register (Read-Only)**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| E3 | E2 | E1 | E0 | R11 | R10 | R9 | R8 |
| R | R | R | R | R | R | R | R |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| R | R | R | R | R | R | R | R |

LEGEND: R = Read only

**Table 7. Result Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15:12 | E[3:0] | R | 0h | **Exponent.**<br>These bits are the exponent bits. Table 8 provides further details. |
| 11:0 | R[11:0] | R | 000h | **Fractional result.**<br>These bits are the result in straight binary coding (zero to full-scale). |

**Table 8. Full-Scale Range and LSB Size as a Function of Exponent Level**

| E3 | E2 | E1 | E0 | FULL-SCALE RANGE (lux) | LSB SIZE (lux per LSB) |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 40.95 | 0.01 |
| 0 | 0 | 0 | 1 | 81.90 | 0.02 |
| 0 | 0 | 1 | 0 | 163.80 | 0.04 |
| 0 | 0 | 1 | 1 | 327.60 | 0.08 |
| 0 | 1 | 0 | 0 | 655.20 | 0.16 |
| 0 | 1 | 0 | 1 | 1310.40 | 0.32 |
| 0 | 1 | 1 | 0 | 2620.80 | 0.64 |
| 0 | 1 | 1 | 1 | 5241.60 | 1.28 |
| 1 | 0 | 0 | 0 | 10483.20 | 2.56 |
| 1 | 0 | 0 | 1 | 20966.40 | 5.12 |
| 1 | 0 | 1 | 0 | 41932.80 | 10.24 |
| 1 | 0 | 1 | 1 | 83865.60 | 20.48 |

The formula to translate this register into lux is given in Equation 1:

$$lux = LSB\_Size \times R[11:0] \quad (1)$$

where:

$$LSB\_Size = 0.01 \times 2^{E[3:0]} \quad (2)$$

LSB_Size can also be taken from Table 8. The complete lux equation is shown in Equation 3:

$$lux = 0.01 \times (2^{E[3:0]}) \times R[11:0] \quad (3)$$

*Figure 92: TI Ambient Light Sensor Brightness Conversion [57]*

## Appendix L: Sink Node to VR Program Communication Protocol

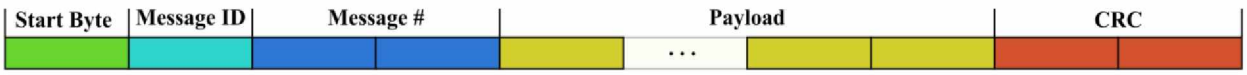


*Figure 93: Sink Node to VR Program Packet Diagram*

Communication between the sink node and the VR program was carried out using a serial COM port set to a baud rate of 115200, 8 bits of data, no parity, 1 stop bit, and no flow control. The communication protocol used a data packet with a variable payload length as seen in Figure 93. This protocol used data packets that ranged between 8 and 14 bytes long. The protocol started with a Start Byte with a value of 0x1E to indicate the start of a packet, followed by a message ID byte, and then a 2-byte message number, followed by a variable length data payload. The data packet was finished with a 2-byte CRC code that was used to verify that the data was not corrupted during transmission. The COM port was asynchronous and full duplex, so the packets were sent as soon as they were ready. Table 27 describes the message types used in this communication protocol. Table 28 shows the button and gesture code used in the "Sensor Node Sensor Data" message.

*Table 27: Sink Node to VR Program Communication Protocol Message Types*

| Message Name | Message ID | Payload | Packet Length | Description |
|---|---|---|---|---|
| Connect/Disconnect | 0x01 | Bytes[1:2]: Connect[0x0000] Disconnect[0xFFFF] | 8 Bytes | Connects or Disconnects the WSN |
| Sensor Node Activity Mode | 0x02 | Bytes[1:2]: Sensor Node ID<br>Bytes[3:4] Fast Mode[0x0000] Slow Mode[0xFFFF] | 10 Bytes | Sets the Mode of the Sensor Node |
| VSN Input Signal | 0x03 | Bytes[1:2]: Sensor Node ID<br>Bytes[3:4] Button or Gesture Code (Table 16) | 10 Bytes | Alerts the Sink Node of a VSN Input |
| Sensor Node Connection | 0xAA | Bytes[1:2]: Sensor Node ID<br>Bytes[3:4] Connected[0x0000] Disconnected[0xFFFF] | 10 Bytes | Indicates that the Sensor Node has connected or disconnected from the WSN |
| Sensor Node Temperature Range | 0xBB | Bytes[1:2]: Sensor Node ID<br>Bytes[3:4] Min Temperature (integer)<br>Bytes[5:6] Max Temperature<br>Bytes[7:8] Current Temperature | 14 Bytes | Transmits the user defined temperature range of the Sensor Node |
| Sensor Node Sensor Data | 0xCC | Bytes[1:2]: Sensor Node ID<br>Bytes[3:4] Temperature (integer)<br>Bytes[5:6] Brightness (integer)<br>Bytes[7:8] Button Statuses | 14 Bytes | Transmits Sensor Data other than the Accelerometer data |
| Sensor Node Accelerometer Packet | 0xDD | Bytes[1:2] Sensor Node ID<br>Bytes[3:4] X Accel (integer)<br>Bytes[5:6] Y Accel (integer)<br>Bytes[7:8] Z Accel (integer) | 14 Bytes | Transmits Accelerometer data |
| Sensor Node Reset | 0xFF | Bytes[1:2] Sensor Node ID<br>Bytes[3:4] 0x0000 | 10 Bytes | Resets statistical values on the Sensor Node |

*Table 28: VSN Button and Gesture Codes*

| VSN Button and Gesture Code | |
|---|---|
| Button #1 True | 0x0100 |
| Button #1 False | 0x01FF |
| Button #2 True | 0x0200 |
| Button #2 False | 0x02FF |
| Button #3 True | 0x0300 |
| Button #3 False | 0x03FF |
| Gesture: Left Punch | 0xFF01 |
| Gesture: Right Punch | 0xFF02 |
| Gesture: Vertical Open | 0xFF03 |
| Gesture: Horizontal Open | 0xFF04 |

## Appendix M: WSN Communication

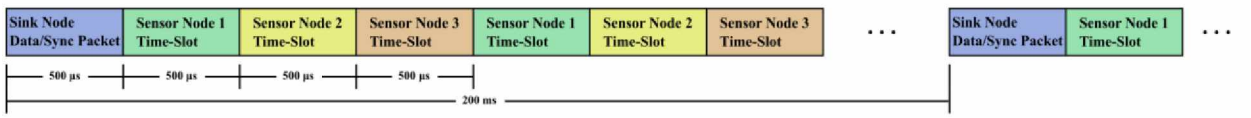


*Figure 94: WSN MAC Protocol*

The MAC protocol used in the WSN was a contention-free time scheduled protocol. This protocol used a 200 ms long frame that began with a data and synchronization packet from the sink node. The frame was divided into four hundred 500 µs time-slots. The first time-slot was reserved for the sink node's packet, and the remaining 399 time-slots were divided sequentially between the three sensor nodes as shown in Figure 94. The sink node was responsible for synchronizing the network by sending a packet that was used by the sensor nodes to synchronize timing events. This packet contained all the data that needed to be transmitted to the sensor nodes in the network. In order to save power, the sensor nodes only took their RF transceivers out of the power-down mode when listening for the synchronization packet or when transmitting data. sensor node data packets were only transmitted when new data was ready at the start of their time-slots, as indicated by several software flags that were set when new data was collected.

This MAC protocol design was used to ensure that the sensor nodes would be able to transmit sensor data to the VR program with a minimal delay while reducing the active time of the transceiver. The frame length of 200 ms was chosen to minimize the cumulative time that the sensor nodes would be listening for packets from the sink node, while also allowing the sensor node mode values to be transmitted frequently enough to allow for an immersive experience in the VR program.

The sink node's data/synchronization packet was four bytes long. The first byte in the packet was the network ID. The remaining three bytes contained the data transmitted to the

sensor nodes from the VSNs. Each of these bytes corresponded to a specific sensor node. The first bit indicated if the sensor node was instructed to be reset. The second bit indicated the mode the sensor node needed to be in. Bits three, four, and five indicated if a gesture was detected by the VSN, and what gesture it was. Bits six, seven, and eight indicated the status of the virtual buttons on the VSN. The sensor nodes used three different data packet types that were all five bytes long: temperature range packet, sensor data packet, and accelerometer data packet. The sink node and sensor node packets are shown in Table 29. These packets did not use any preamble or error detection mechanisms as the nRF24L01+ transceiver automatically attached, analyzed, and removed the needed overhead without any intervention from the microcontroller.

*Table 29: WSN Communication Packets*

| Source | Packet Type | Payload | Packet Length |
|---|---|---|---|
| Sink Node | Data/Synch | Byte[1]: Network ID<br>Byte[2]: VSN to Sensor Node 1 Data<br>Byte[3]: VSN to Sensor Node 2 Data<br>Byte[4]: VSN to Sensor Node 3 Data | 4 Bytes |
| Sensor Node | Temperature Range | Byte[1]: Sensor Node ID and Packet Type<br>Bytes[2:3]: Minimum Temperature (integer)<br>Bytes[4:5]: Maximum Temperature (integer) | 5 Bytes |
| Sensor Node | Sensor Values | Byte[1]: Sensor Node ID and Packet Type<br>Bytes[2:3]: Current Temperature (integer)<br>Byte[4]: Percent Brightness (8-bit integer)<br>Byte[5]: Button Statuses | 5 Bytes |
| Sensor Node | Accelerometer Values | Byte[1]: Sensor Node ID and Packet Type<br>Byte[2]: Signs of Accelerometer Values<br>Byte[3]: Absolute Value of X Axis Acceleration<br>Byte[4]: Absolute Value of Y-Axis Acceleration<br>Byte[5]: Absolute Value of Z-Axis Acceleration | 5 Bytes |